Inhaltsverzeichnis

Projektdokumentation



Thema: Aufbau einer Firewall mit Grenznetz und Bastion Host
mit der Möglichkeit zur Remoteadministration

Team: Michael Bruckmann
Olaf Plaggenborg

Beginn: 07.Februar 2005

Ende: 09.05.2005

1 Einleitung

Das Internet bzw. Netzwerk bietet aufgrund der schnellen Entwicklung von PC-Hardware und Softwareprogrammen in der heutigen Zeit, ungeahnte Möglichkeiten der Kommunikation und des Datenaustausches zwischen verschiedenen Standorten, externen Dienstleistern und sonstigen Quellen wie WWW oder FTP. Angestellte können von zu Hause aus im Firmennetz arbeiten, Aussenstellen haben die Möglichkeit auf interne Datenbanken zurück zu Greifen, Server unter externer Wartung sind online ohne der Anreise eines Technikers konfigurierbar, und aktuelle Programme und Treiber liegen auf öffentlichen FTP-Servern zum herunterladen bereit.

Nur was ist eigentlich mit der Sicherheit ?

Wenn über Computersicherheit gesprochen wird, hört man oft: „Ich habe keine Probleme. Meine Daten sind weder wertvoll noch geheim, und falls jemand meine Festplatte löscht, habe ich kein Problem damit, alles neu zu installieren.“ Wenn dies der Fall ist, so brauchen Sie eigentlich keine Sicherheit. Oder ?

In der Regel ist es aber nicht ganz so einfach. Oft liegen der Vorstellung, nicht von diesen Problemen betroffen zu sein, falsche Annahmen zugrunde. Aber auch hier gilt wie fast überall im Leben: „Unwissenheit schützt vor Strafe nicht“

Bei der Suche nach einem geeigneten Projekt-Thema, wollten wir uns deshalb gerne mit dem Thema Computersicherheit beschäftigen. Aus diesem Grund haben wir uns für einen Projektvorschlag der Berufsbildenden Schulen Friedenstraße entschieden, ein Firewall-System für diese Schule zu entwickeln. Die Firewall sollte auf einem extra Server mit dem freien Betriebssystem Linux (Open-Source) realisiert werden und zur zusätzlichen Sicherheit eine Festplattenspiegelung mit sogenanntem Software-RAID durchgeführt werden. Als Firewallarchitektur diente Iptables/Netfilter die im Linux Kernel seit Version 2.4 vorhanden sind.

Somit traten wir hoch motiviert diese Aufgabe an. Zuerst galt es für dieses Projekt unser Wissen im Bereich PC-Hardware, Linux, und Netzwerktechnik aufzufrischen, welches wir während der bisherigen Technikerausbildung vermittelt bekommen haben. Die Schwierigkeit bestand darin sich mit dem völlig neuen Thema Firewall bzw. Iptables auseinander zusetzen. Diese Regeln stellten sich für uns am Anfang doch recht kryptisch dar. Aber auch hier war nach einiger Zeit „Licht am Ende des Tunnels“. Weitere umfangreiche Bereiche stellte zum einen der Umgang mit dem Tool Firewall Builder (Anleitung nur in Englischer Sprache) dar und zum anderen ein Aussagekräftiger Testaufbau zur Abnahme der Funktionen.

Für unser Team war dieses Projekt eine große Herausforderung und wir machten uns mit viel Einsatz und Engagement an die uns gestellte Aufgabe, um diese zu einem positiven Abschluss zu bringen.

2 Projektdokumentation

2.1 Aufgabenstellung

Unser Aufgabe bestand darin für das Netzwerk der Berufsbildenden Schulen Friedenstraße eine Firewall mit Grenznetz und zusätzlichem dual-homed Bastion-Host aufzubauen und zu testen. Außerdem sollte aus Sicherheitsgründen Software RAID (Festplattenspiegelung) eingesetzt werden und zur Administration Webmin verwendet werden. Damit wurde eine Fernwartung der Firewall vom Administrator gewährleistet. Für die Installation des Servers verwendeten wir das Betriebssystem Linux (Fedora Core 2). Die Firewall wurde mithilfe von Netfilter realisiert, iptables diente dazu dieses Subsystem zu konfigurieren. Damit wir die iptables Befehlsstrukturen in der Praxis besser umsetzten konnten, verwendeten wir für unser Projekt das graphische Hilfstool Firewallbuilder.

2.1.1 Aufgaben des Projektteams

- ☑ Linux Installation (mit RAID)
- ☑ Grafische Oberfläche (Extra Partition, muss von root händisch gemountet werden)
- ☑ Keine Überflüssigen Dienste und Softwarepakete dürfen installiert sein
- ☑ Einarbeitung in die Technik einer Firewall (iptables)
- ☑ Einarbeitung in die Software Firewallbuilder
- ☑ Erstellung der Firewall Regeln
- ☑ Dokumentation und Test aller Schritte
- ☑ Schulung Administratoren
- ☑ Präsentation

2.1.2 Hardware

Diese Server-Hardware wurde uns für die Projektarbeit von den Berufsbildenden Schulen zur Verfügung gestellt:

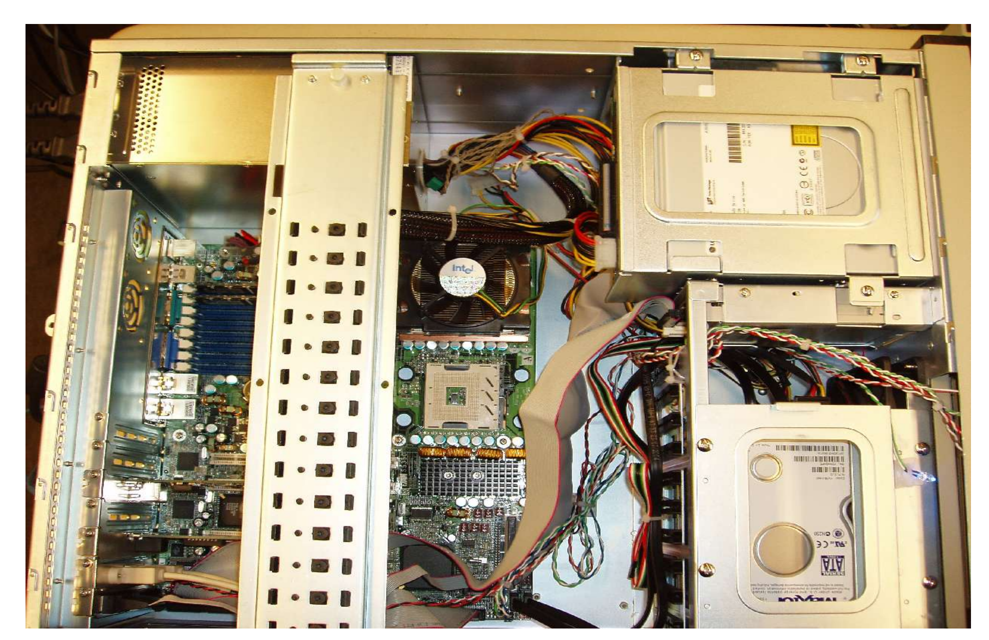


Abb. 2.1.2: Firewall Rechner Innenleben

- ☑ Servergehäuse 19“ Rack Einheit 4HE
- ☑ Motherboard Tyan S5350 mit 2x CPU Xeon 2800 MHz FSB 800
- ☑ 1 GB ECC PC 333 RAM
- ☑ Grafik on Board
- ☑ 2 x 400 W redundantes Netzteil
- ☑ LG GSA 4160 DVD Brenner
- ☑ Diskettenlaufwerk 3,5“
- ☑ 2 x Maxtor SATA Festplatte 120 GB
- ☑ 2 x GB LAN on Board
- ☑ 1 x Intel 10/100 PCI (Standleitung)
- ☑ 1 x Intel 1 GB LAN

2.1.3 Software

Folgende Software sollte verwendet werden bzw. wurde zur Verfügung gestellt:

Linux : Fedora Core 2 C (inklusive aller Updates); wurde gestellt

Webmin*[1]

Tool zur Administration von z.B. Netzwerk, Server, Hardware usw.

Download: http://www.webmin.com/

Firewall Builder

GUI zur Erstellung der Firewall-Regeln

Download: http://www.fwbuilder.org/

Zusätzlich wurden noch weitere nützliche Tools eingesetzt, die bei der Fehlersuche in der Firewall oder anderen Netzwerk-Funktionen recht hilfreich waren.

ethereal

Network-Protokoll-Analyzer

Download: http://www.ethereal.com/

nmap

Netzwerk-Mapper (Port-Scanner)

Download: http://www.insure.org/nmap

ping

Sendet einen ICMP ECHO_REQUEST an festgelegte Hosts

traceroute

Gibt die Route aus, die die Pakete zu einem bestimmten Host nehmen

Download: http://www-nrg.ee.lbl.gov/

Nessus

Sicherheits-Scanner

Download: http://www.nessus.org/intro.html

*[1] Hier eine kurze Erklärung zu Webmin. In unserer Projektarbeit haben wir dieses Tool nur zur Einrichtung des Netzwerkes benutzt und für ein paar kleinere Funktionen während unserer Testphasen. Im späteren Schulbetrieb wird es für die Remoteadministration verwendet. Deshalb wollen wir dieses Tool nur kurz in unser Dokumentation erwähnen.

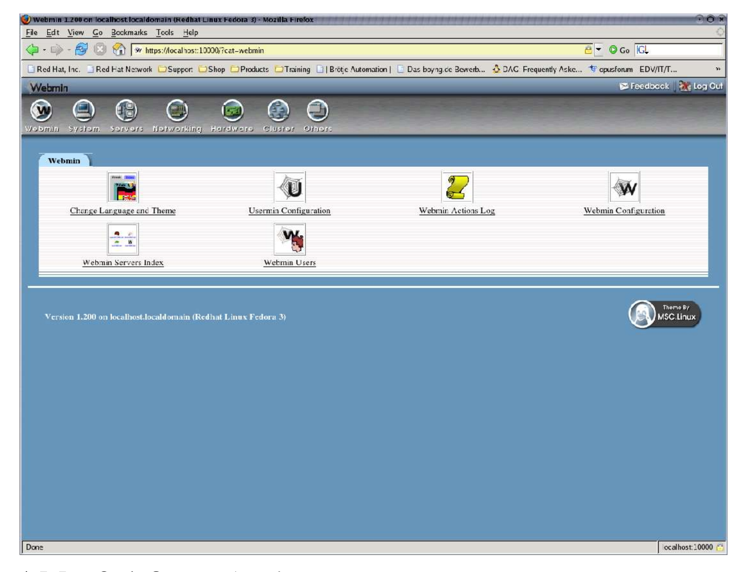


Abb. 2.1.3: Webmin

Die Administration von Linux-Servern besteht für viele Anwender aus Kommandozeilen und kryptischen Konfigurationsdateien. Eine einfach und intuitiv zu bedienende einheitliche Oberfläche wie bei Windows oder MacOS fehlt bei den meisten Distributionen oder eignet sich nicht für die Fernadministration.

Genau an dieser Stelle setzt Webmin an. Diese Software bietet eine grafische Oberfläche zum Verwalten und Konfigurieren der gebräuchlichsten Dienste unter Linux/Unix. Über Webmin ist es auch unter Linux möglich Dienste wie SAMBA, Apache, NFS oder FTP über eine Grafische Oberfläche zu konfigurieren. Die klassische Systemverwaltung, wie zum Beispiel die Benutzer- und Gruppenverwaltung, ist ebenfalls über Webmin möglich.

Da Webmin im Webbrowser läuft, kann er auch zur Remote-Administration eingesetzt werden. Ein besonderes Feature ist hierbei, dass Webmin auch verschlüsselte Verbindungen per SSL aufbauen kann. Damit ist dann auch die sichere Fernwartung über das Internet realisierbar. Oder wie für unsere Projektarbeit gefordert, über das Netzwerk.

2.1.4 Netzwerk

Hier sehen Sie wie das gesamte Netzwerk nach Fertigstellung der Projektarbeiten aussehen soll.

BBS-FIRE als Server stellt unsere Hauptprojektarbeit dar, einen eigenständigen Server der als Firewall arbeitet. Zusätzlich dazu noch die Integration einer zusätzlichen Firewall auf dem SUPER Server (Terminal Server Projektarbeit), welcher bis auf die eben erwähnte Firewall von einer anderen Projektgruppe erstellt wurde.

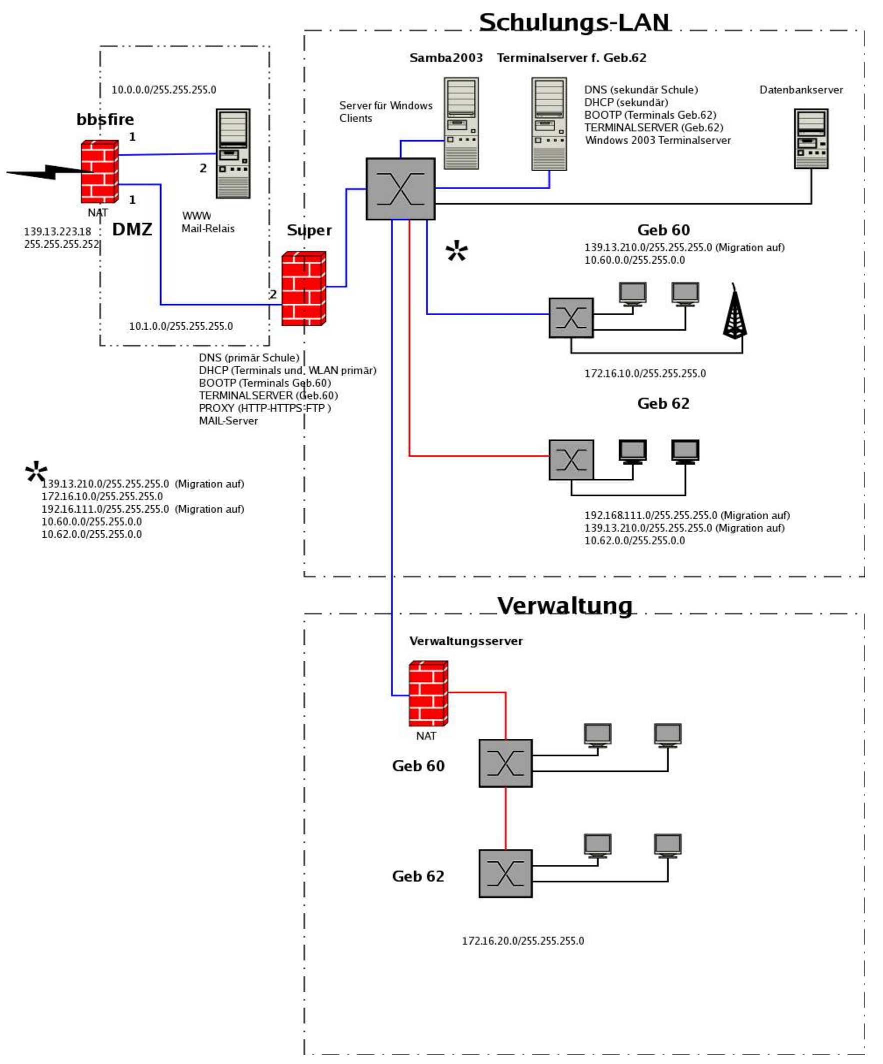


Abb. 2.1.4: Das Schulnetz

2.1.5 Dateisystemstruktur (Partitionierung)

Für unsere Projektarbeit wurde ein Software RAID System*[2] (RAID 1) eingesetzt. Die Festplatten (2 x 120 GB) wurden wie folgt partitioniert:

	Mount-Point	Filesystem	Größe (ca.)
/dev/md0	/	ext2	30 GB
/dev/md1	/usr/X11R6	ext2	30 GB
/dev/md2	MINIMAL INSTALLATION	ext2	30 GB
/dev/md3	swap		2 GB
/dev/md4	/opt	ext2	28 GB

*[2] Ein **RAID**-System (Abk. **R**edundant **A**rray of **I**nexpensive / **I**ndependent **D**isks) dient zur Organisation von mehreren Festplatten bei einem Computer.

Durch die Verwendung von RAID-Systemen kann man die Betriebssicherheit, Leistung und/oder Kapazität von Massenspeichern erhöhen.

Dazu gibt es verschiedene Möglichkeiten, die man als RAID-Levels definiert hat.

RAID-Systeme erfordern bei der Einrichtung mehr Aufwand, während sie sich für Benutzer nicht von herkömmlichen Massenspeichern unterscheiden und können durch Controller mit RAID-Funktionalität (Hardware-RAID) oder auf konventionellen Controllern mit speziellen Treibern (Software-RAID) realisiert werden. In unserem Projekt haben wir Software-RAID im Level 1 verwendet, dieses bedeutet:

RAID 1 (Spiegelung)

RAID 1 bietet Redundanz der gespeicherten Daten, da diese immer auf mindestens zwei Festplatten in identischer Form vorliegen (Spiegelung). Fällt eine Platte aus, kann eine andere für sie einspringen.

2.2 Projektplanung

2.2.1 Projektzeitraum

Projekt-Beginn: 07.Februar 2005

Projekt-Ende: 09.Mai 2005

2.2.2 Projektteam

Michael Bruckmann

Olaf Plaggenborg

2.2.3 Projektbetreuer

Herr Appenzeller (Berufsbildenden Schulen Friedenstraße Wilhelmshaven)

2.2.4 Projekthilfsmittel

Jeder Mitarbeiter des Projektteams benutzte sein eigenes Notebook. Die Berufsbildenden Schulen Friedenstraße stellte uns einen neuen Server und Räumlichkeiten zur Nutzung zur Verfügung. Somit hatten wir die Möglichkeit, vor Ort auftretende Fragen und Probleme direkt mit dem Projektbetreuer Herrn Appenzeller zu erläutern, zu überarbeiten und zu lösen.

2.2.5 Projektkosten

Bei unserer Projektarbeit, „Aufbau einer Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration“, entstanden uns keine Kosten.

2.3 Aufgabenverteilung

Michael Bruckmann

☑ Installation Linux (Fedora Core 2)

☑ Einarbeitung in die Technik einer Firewall (iptables)

☑ Einarbeitung in die Software Fwbuilder

☑ Erstellung der Firewall Regeln

☑ Test der Firewall

☑ Dokumentation

☑ Präsentation

Olaf Plaggenborg

☑ Installation Linux (Fedora Core 2)

☑ Einarbeitung in die Technik einer Firewall (iptables)

☑ Einarbeitung in die Software Fwbuilder

☑ Erstellung der Firewall Regeln

☑ Test der Firewall

☑ Dokumentation

☑ Präsentation

2.3.1 Kurzbeschreibung des Projekts

Der Startschuss für unser Projekt, war der 07.02.05. Wir bekamen unseren Server, der bereits zusammen gebaut, in einem extra Klassenzimmer stand und das Betriebssystem Fedora Core 2C auf einer DVD. Die Software, die wir sonst noch brauchten, besorgten wir uns im Laufe der Projektarbeit aus dem Internet. Außerdem erhielten wir von unserem Projektleiter Herrn Appenzeller eine allgemeine Projekt-Einweisung.

Am ersten offiziellen Projekttag, den 09.02.05 begannen wir voller Tatendrang mit unserer Arbeit, wir installierten Linux im RAID Level 1, was keine größeren Probleme machte. Nach dem Neustart des Systems startete der Bootloader-Grub aber nicht, was zur folge hatte, dass wir das System nicht starten konnten. Unser erstes Problem war geboren. Fehlersuche war angesagt, also durchsuchte einer von uns das Internet und der zweite studierte die Beschreibung vom Mainboard.

Wir brachen den RAID Verbund auf, aber ohne Erfolg das System wollte nicht starten. Es mußte wohl an Linux liegen, dachten wir und machten ein komplettes Fedora Update inklusive Kernel. Aber auch das brachte nicht den gewünschten Erfolg. Schließlich kamen wir zu dem Schluß, es muß am Mainboard liegen. Aus diesem Grund beschlossen wir ein BIOS Update auf Version 1.03 durchzuführen, außerdem partitionierten wir die Festplatte neu im RAID Level 1 und installierten Fedora Core 2C neu. Siehe da: Voller Erfolg!

- System bootet
- Bootloader Grub startet
- Linux startet,

das System startet.

Jetzt konnten wir den weiteren Ablauf erstellen und endlich die weitere Software installieren(Webmin v1.180-1, usw.). Wir installierten wieder einen neuen Kernel und führten dann das aktuelle Fedora Core 2C Update durch.

Ab der zweiten Woche versuchten wir dann das Netzwerk einzurichten. Wir installierten den LAN Treiber für unsere beiden Onboard Netzwerkkarten und setzten eine zusätzliche Karte ein, da wir drei Interfaces für unser Projekt benötigten und schon hatten wir unser nächstes Problem. Linux erkannte nur eine der beiden Onboard Karten. Also begaben wir uns erneut auf Fehlersuche, wir bauten die zusätzliche Karte aus, ohne Erfolg. Dann installierten wir das alte BIOS v1.02 und starteten mit dem Betriebssystem Knoppix und später auch noch mit Fedora Core 3, ohne Erfolg.

Unser Projektleiter hatte dann die zündende Idee, dass wir es mit einem CMOS Reset probieren sollten. Gesagt getan, neuer Versuch mit Fedora Core 3 und die Netzwerkkarten wurden beide erkannt. Wir änderten das BIOS auf Version 1.03 und setzten die dritte Netzwerkkarte ein. Es folgte ein Neustart und wieder war eine Onboard Karte verschwunden. Ratlosigkeit machte sich breit.

Also fingen wir von vorne an, Zusatzkarte heraus nehmen, altes BIOS v1.02 wieder installieren, Fedora Core 3 neu booten und beide Karten waren wieder da. Vorsichtshalber konfigurierten wir die beiden Karten mit dem Tool Webmin, machten einen Neustart und die beiden Interfaces wurden gestartet. Am darauf folgenden Tag steckten wir dann die Zusatzkarte ein, womit wir wieder bei unserem alten Problem angelangt waren, die eine Onboard Karte streikte.

Kurz gesagt, unter Verwendung von BIOS v1.03, war es nicht möglich beide Onboard Karten und eine zusätzliche Netzwerkkarte zu betreiben. Also mußten wir improvisieren und besorgten uns eine Netzwerkkarte (INTEL 1G), mit der wir dann unser Problem beheben konnten. Somit wurde das Ziel erreicht, wir hatten unsere drei Interfaces, die wir für unser Projekt benötigten und richteten sie mit Webmin ein.

Nun konnten wir mit unserem eigentlichen Thema beginnen, dem Firewall Builder. Wir besorgten uns die aktuelle Version 2.0.6 und installierten sie. Jetzt war es an der Zeit, sich mit der Materie Iptables und Fwbuilder, anhand von Büchern und dem Internet auseinandersetzen. Im Internet wurden wir schnell fündig, der Fwbuilder als Benutzerhandbuch zum Downloaden. Einziges Problem die Beschreibung war komplett auf Englisch und Deutsche Hilfen gab es kaum. Somit mußten wir natürlich die für uns wichtigen Teile, während der Arbeit mit dem Fwbuilder, ins Deutsche übersetzen. Nebenbei beschäftigten wir uns, unter Zuhilfenahme von Büchern, noch mit Firewalls (allgemein) und Iptables, was einige Zeit in Anspruch nahm.

Wir machten unsere ersten „Gehversuche“ mit dem Fwbuilder und versuchten eine eigene Firewall für Testzwecke zu erstellen. Dazu gehörte natürlich auch das Einrichten der gültigen IP-Adressen für eth0, eth1 und eth2, einen Verbindungstest zwischen unseren Notebooks und dem Firewall Rechner. Als nächstes sollte eines der Notebooks den Webserver simulieren, was wir mit dem Apache Webserver Tool realisierten.

Ab dem 14 Projekttag haben wir dann begonnen, uns Gedanken zu machen über die eigentlichen Firewalls. Zu diesem Zweck stellten wir die erforderlichen Grundvoraussetzungen her, wie z.B. die Host Rechner und die Netzwerke im Schulnetz. Während der Arbeit mit dem Fwbuilder notierten wir alle Schritte, für die spätere Erstellung der Firewall Builder Anleitung und des Benutzerhandbuchs. Dies war nötig um dem Administrator die Arbeit mit dem Fwbuilder und den beiden Firewalls zu erleichtern.

Es kam natürlich gelegentlich zu Fehlschlägen, was die Firewall Regeln angeht, aber unsere Bücher, das Internet und vor allem Herr Appenzeller halfen uns immer wieder diese Hürden zu nehmen und uns auf den richtigen Weg zu bringen. Es war ein ständiges hin und her was die Regeln angeht, da es immer wieder zu unvorhergesehenen Schwierigkeiten zwischen den einzelnen Netzen (Webserver, Proxy und Terminalserver, Firewall Rechner) kam. Es ging sogar soweit, dass der Proxy aus dem Terminalserver herausgenommen werden mußte und seinen Dienst seit dem auf dem Firewall Rechner verrichtet.

Eine genauere Beschreibung zu den Regeln und dem Fwbuilder folgt in den nächsten Kapiteln.

Um das Zugreifen für Eindringlinge so schwierig wie möglich zu gestalten, sollten wir noch die Dienste auf ein Minimum reduzieren und die Grafik abmounten, was wir auch relativ schnell in den Griff bekamen. Wenn es dennoch ein Eindringling geschafft hätte unsere Firewall ausser Gefecht zu setzen, hätte man noch eine Minimal Installation gehabt auf der die Firewall zusätzlich installiert ist, um das System sofort wieder starten zu können, ohne große Ausfallzeiten zu haben.

Womit wir bei der minimal Installation wären. Nach einigem hin und her überlegen, kamen wir zu dem Schluß, dass es mehr Sinn macht von vornherein die Minimale zu benutzen, da auf dieser gar keine Grafik vorhanden ist und die Dienste immer auf ein Minimum beschränkt sind, was die Sache auf den ersten Blick deutlich einfacher Aussehen läßt. Also machten wir uns an die Arbeit, eine Partition frei geben für die Minimale Installation und dann aufspielen von Fedora Core 2C Minimal. Von unseren drei Netzwerkkarten wurden zwei erkannt, was kurzfristig Panik auslöste. Beim Neustart von Linux bot uns die Hardware-Erkennung KUDZU die dritte Netzwerkkarte an und wir konnten diese einrichten. Zur Sicherheit haben wir die Interfaces, eth0, eth1 und eth2 ihren eigentlichen IP`s zugeordnet, um bei der Firewall später kein Risiko einzugehen. Sie sollten bei beiden Installationen identisch sein.

Jetzt galt es noch die Firewall *(bbsfire.fw)* bei der Minimal Installation aufzusetzen, was vom Superserver aus geschehen sollte, da auch hier der Fwbuilder installiert war. Die Version 2.0.6 bietet das Installieren von einem zum anderen Rechner an, sofern die alten Regeln dieses natürlich zulassen oder der SSH Zugriff dieses verbietet.

Bei abschließenden Tests mit Herrn Appenzeller, ob die Regeln auf *bbsfire* und *Super* auch wirklich den gewünschten Effekt bringen, brauchten wir nur noch geringfügig an unseren Regeln Verbesserungen bzw. Veränderungen einarbeiten, die sich bei den Tests ergaben, um dann unseren praktischen Projektteil zu einem positiven Abschluß zu bringen. Von da an konnten wir uns ganz der Dokumentation widmen, was natürlich noch eine erhebliche Zeit in Anspruch nahm.

2.4 Wer braucht eine Firewall?

2.4.1 Sind Cracker Genies?

Sie hätten vermutlich kaum das Bedürfnis, sich unter Millionen von Rechner im Internet Ihren vorzunehmen, wenn dies so wäre. Statt dessen würden sie sich mit Rechnern von Regierungen und Banken beschäftigen, die eine viel größere Herausforderung und ein finanziell lohnenswertes Ziel bieten.

Diese Sorte Cracker ist extrem selten. Gelangweilte Teenager sind das Hauptproblem, die Angriffsprogramme nutzen, deren Funktionsweise sie oftmals nicht einmal verstehen. Solche Angreifer werden daher auch als SCRIPT KIDDIES bezeichnet, da tiefer gehende Computerkenntnisse mit solchen Programmen nicht erforderlich sind.

Genau deswegen ist das Risiko, das Ziel eines Angriffs zu werden auch größer, als man zunächst annehmen sollte. Es existiert eine große Anzahl an Angreifern, die automatisierte Werkzeuge besitzen, mit denen sie in kürzester Zeit riesige Bereiche im Internet untersuchen und schlecht gesicherte Rechner automatisch angreifen können.

2.4.2 Durch Cracker gelöschte Daten einfach neu erstellen?

Anwender gehen davon aus, dass sie keine wichtigen Daten besitzen oder ihre Daten im Notfall neu erstellen können, indem sie auf vorhandene Unterlagen und Notizen zurückgreifen. Wenn man seinen Rechner aber beruflich nutzt kann so eine Einstellung fatal sein. Man kann vermutlich die meisten Dateien neu erstellen. Aber wieviel Arbeitszeit kostet das? Was ist wenn ein Angriff Abends erfolgt und wichtige Dokumente oder gar eine Präsentation für den nächsten Tag verloren gehen?

Hier ein Tipp! Einmal in Ruhe den Inhalt der Festplatte durch gehen und sich fragen, was es bedeuten würde, wenn die dort abgelegten Dateien verloren gehen würden. Was wäre wenn man das Netz als Informationsquelle nutzt und die komplette Lesezeichenliste des Browsers, die man Monatelang gesammelt hat, müßte neu erstellt werden.

Das beste Mittel gegen Datenverlust ist sicher ein tägliches Backup. Darüber hinaus sollte man von vornherein die Wahrscheinlichkeit verringern, dass dieser Ernstfall eintritt. In Bezug auf Angriffe aus dem Internet kann dies durch den Aufbau einer Firewall geschehen.

2.4.3 Vorgehensweise von Crackern

Angenommen, es gäbe ein kleines Softwareunternehmen, das Software für mittelständische Unternehmen schreibt und jeden Abend werden Backups angefertigt und an einem sicheren Ort hinterlegt.

Wollte nun ein Konkurrent der Firma einen Schaden zufügen, durch löschen der Festplatte, so wäre der Schaden eher gering. Es würde eine Verzögerung von wenigen Tagen entstehen durch neues aufsetzen des Rechners. Der Recher würde danach aber sicherlich besser gesichert werden, um Angreifern keine 2. Chance zu bieten.

Anders sieht es aus, wenn sich der Angreifer Zugang verschafft und nur kleine Änderungen vornimmt, z.B. kleine Fehler in Programme einbaut, die gerade entwickelt werden. Wenn so ein Angriff geschickt ausgeführt wird, ist die Chance groß, dass es erst sehr spät entdeckt wird und es wird nahezu unmöglich, alle Fehler zu finden. Auch Backups helfen wenig, wenn das genaue Datum nicht bekannt ist. Sollte man aber doch ein „sauberes“ Backup finden, so ist es meist zu alt, als das es viel nützt.

Glücklicherweise sind solche Fälle eher selten, in denen Cracker so vorgehen.

2.4.4 Was wollen Cracker?

Eigene Dateien sind vermutlich eher uninteressant für den durchschnittlichen Cracker. Zwei andere Dinge werden ihn vielmehr interessieren.

Zum einen seine Ressourcen. Einen Rechner, den niemand mit dem Cracker in Verbindung bringen kann, könnte er für viele Dinge nutzen. Er könnte Server einrichten um mit Gleichgesinnten Daten auszutauschen. Auch sind fremde Rechner als Ausgangsbasis für eine Angriff auf Drittrechner viel besser geeignet.

Als zweites ist es für einen Cracker interessant, die Anwender zu beobachten, die einen Rechner benutzen. Er kann Paßwörter belauschen. Kauft man online ein kann er sogar die Kreditkartennummer abfangen und für eigene Einkäufe verwenden. Hierzu wird er Programme installieren, die Tastatureingaben oder Pakete im lokalen Netz belauschen, um nicht ständig zugegen zu sein. Er wird diese gesammelten Informationen dann bei einem späteren „Besuch“ von dem jeweiligen Rechner ziehen. Das schreiben solcher Programme ist nicht wirklich schwierig, aber der durchschnittliche Cracker wird hierfür verfügbare Programme aus dem Internet verwenden.

2.5 Was ist eine Firewall?

Eine gängige Maßnahme, um sich vor den Gefahren des Internets zu schützen, besteht darin, zwischen die Rechner im eigenen Netz und das Internet eine Firewall zu installieren. Diese kann man sich wie das Burgtor einer mittelalterlichen Stadt vorstellen. Es ist der einzige Zugang zur Stadt, die ansonsten von allen Seiten durch hohe Mauern geschützt ist. Um in die Stadt gelangen zu können, muß ein Besucher an Wachen vorbei, die ihn nach seinen Papieren befragen. Erst wenn er ihnen Rede und Antwort gestanden hat, darf er die Stadt betreten.

Eine Firewall schützt in ähnlicher Weise den Zugang zum lokalen Netz. Jeglicher Verkehr muß sie passieren und wird von ihr untersucht. Damit kann ein Großteil der möglichen Angriffe schon abgefangen werden, bevor sie ihr Ziel erreichen. Allerdings ist dieser Schutz nicht vollkommen. Gerade der Fall der Stadt Troja hat eindrucksvoll bewiesen, welche Folgen es hat, wenn bei der Untersuchung des hereinkommenden Verkehrs eine falsche Entscheidung getroffen wird. Im folgenden werden wir sehen, wie uns eine Firewall beim Schutz unserer Rechner helfen kann und was mit anderen Mitteln erreicht werden muß.

2.5.1 Was eine Firewall kann

Wie ein Stadttor alle Angriffe auf einen stark gesicherten Punkt bündelt, an dem man den Großteil seiner Kräfte konzentriert, so stellt auch die Firewall eine Möglichkeit dar, Angriffe an einer definierten Stelle abzufangen.

Besitzt man nur einen Rechner, mit dem man Dienste im Internet nutzt, so kommt man nicht umhin, ihn so zu konfigurieren, dass er keine Schwachstellen aufweist, die ein Angreifer ausnutzen kann, um unberechtigt Zugang zu ihm zu erlangen. Besitzt man aber hundert Rechner, so wird es schwierig, alle immer auf dem neuesten Stand zu halten, Patches gegen Sicherheitslücken einzuspielen und immer darauf zu achten, dass keine unsicheren Dienste auf ihnen installiert sind. Ehe man sich versieht, hat schon ein Benutzer eine Freigabe auf Laufwerk C erstellt, deren Paßwort nicht vorhanden oder leicht zu erraten ist. Oder es richtet sich eine .rhosts-Datei ein, die den Zugang ohne Paßwort von einem Rechner im Internet erlaubt. Werden die Rechner gar von verschiedenen Personen administriert, so kann man darauf wetten, dass die einzelnen Rechner unterschiedlich sicher konfiguriert sind.

In so einem Fall kann man die Sicherheit des Systems verbessern, indem man an zentraler Stelle dafür sorgt, dass Angriffe abgefangen werden, bevor sie ein möglicherweise gefährdetes System erreichen. So reduziert man die Angriffspunkte von 100 auf einen und kann für dieses System eine hieb- und stichfeste Konfiguration entwickeln.

Eine Firewall untersucht den Datenverkehr und lässt nur die Datenzugriffe zu, die vorher definierten Regeln genügen. So ist es z.B. üblich, Anfragen von Rechnern im lokalen Netz an Rechner im Internet zu erlauben,nicht aber umgekehrt. D.h., wenn ein Rechner im lokalen Netz z.B. eine Webseite von einem Server im Internet anfordert, so wird die Antwort (die Webseite) von der Firewall entgegengenommen und in das lokale Netz weitergeleitet. Pakete von Rechnern im Internet werden aber nicht durchgelassen, wenn sie nicht zuvor explizit von einem Rechner im lokalen Netz angefordert wurden. Schon diese Regel verhindert, dass ein Angreifer auf möglicherweise schlecht gesicherte Dienste von Rechnern im lokalen Netz zugreift.

Auch kann man definieren, dass ein Benutzer auf bestimmte Dienste zugreifen darf, auf andere aber nicht. So kann man z.B. verhindern, dass er aus Unkenntnis Protokolle benutzt,bei denen das Paßwort im Klartext übertragen wird.

Wird auf Webserver zugegriffen, bieten einige Firewalls auch die Möglichkeit, unerwünschte Inhalte zu filtern. So kann man z.B. das Laden von Werbebannern unterdrücken, aktive Inhalte aus Webseiten beim Herunterladen entfernen und das Senden von Cookies verhindern.

Kommt es zu verdächtigen Zugriffen auf die eigenen Rechner, so bietet eine Firewall die Möglichkeit, diese zu protokollieren und für eine spätere Auswertung zu speichern. Ohne eine Firewall könnte dies nur auf den Zielsystemen geschehen und würde bedeuten, entweder auf jedem Rechner eigene Auswertungen durchzuführen oder eine Zusätzliche Software zu installieren, die die Systemprotokolle aller Rechner an einer zentralen Stelle zusammenführt. Hinzu kommt, dass nicht alle Betriebssysteme die gleichen Protokollierungsmöglichkeiten besitzen. Auch sind die einzelnen Protokollierungsmechanismen untereinander nicht immer kompatibel.

Auch zur Verringerung der Netzlast kann eine Firewall eingesetzt werden. Laufen alle Zugriffe über einen zentralen Rechner, so bietet es sich an, an dieser Stelle einen Mechanismus zu installieren, der es erlaubt, häufig heruntergeladene Inhalte zwischenzuspeichern (Cachender Proxy). Fordern dann mehrere Benutzer z.B. dieselbe Webseite an, so braucht diese nur für den ersten von ihnen tatsächlich aus dem Internet heruntergeladen zu werden.

Alle weiteren Nachfragen werden aus dem Zwischenspeicher bedient. Dies macht den Zugriff zwar nicht zwangsläufig sicherer, kann aber die Netzlast um 40-60 Prozent verringern.

Schliesslich kann man auch noch kompliziertere Strukturen aufsetzen. Wie mittelalterliche Burgen mehrere Burghöfe besaßen, die ein Angreifer überwinden mußte, bevor er vor dem eigentlichen Wohnhaus des Hausherrn anlangte, so kann auch eine Firewall-Architektur aus mehreren Netzen mit unterschiedlichem Schutzbedarf bestehen. So sind Server, die aus dem Internet zugreifbar sein sollen, einem deutlich höheren Risiko durch Angriffe ausgesetzt und könnten bei einer

Kompromittierung als Ausgangsbasis für weitere Angriffe gegen das lokale Netz genutzt werden. Bringt man diese aber in einem eigenen Netz unter, einer so genannten Demilitarized Zone oder kurz DMZ, so verhindert man, dass ein erfolgreicher Angriff auf einen öffentlichen Server dem Angreifer den Zugriff auf die Arbeitsplatzrechner erleichtert.

2.5.2 Was eine Firewall nicht kann

Obwohl eine Firewall ein wichtiges Werkzeug ist, um die Sicherheit eines Rechners zu erhöhen, so ist sie doch nicht das eine Werkzeug, das ganz allein alle Probleme beseitigt. Eine Firewall ist nur ein Baustein in einer ganzen Reihe von technischen und organisatorischen Maßnahmen, die nötig sind, wenn man einen brauchbaren Schutz von Systemen erreichen will.

Eine Firewall wird z.B. nicht vor Angriffen schützen, die aus dem eigenen Netz heraus ausgeführt werden. Eine gängige Faustregel besagt, dass 80% aller computergestützten Delikte von Insidern begangen werden. Wenn sich eigene Anwender dazu entschliessen, Angriffe auf die eigenen Server durchzuführen, Daten zu löschen oder fremde Dateien auszuspionieren, so ist selbst die beste Firewall absolut wirkungslos.

Dazu ist nicht einmal böser Wille nötig. Es reicht, dass ein Anwender Software herunterlädt oder von zu hause mitbringt und auf einem der Rechner installiert. War diese mit Viren verseucht oder handelt es sich um einen Trojaner, so kann man unwissentlich großen Schaden anrichten. Auch ein Laptop ohne installierten Virenschutz, der von zu hause mitgebracht wird, kann zu einer Verseuchung des ganzen Netzes führen.

Auch kann eine Firewall keinen Netzverkehr kontrollieren, der nicht über sie geleitet wird. In größeren Netzen kann es durchaus schon einmal vorkommen, dass Angestellte, die mit dem gebotenen Internet-Zugang unzufrieden sind, ein eigenes Modem an ihren Computer anschließen. Damit umgehen sie natürlich alle Schutzmaßnahmen, die in der Firewall realisiert werden.

Um solchen Risiken zu begegnen, sind technische Maßnahmen weitgehend wirkungslos. Hier hilft einzig, organisatorische Regeln, POLICIES genannt, aufzustellen, die den Umgang des Benutzers mit den von einem selbst betreuten System regeln. Darüber hinaus ist es auch nötig, Aufklärungskampagnen durchzuführen, die diese Regeln allgemein bekannt machen, und im schlimmsten Fall auch durch disziplinarische Maßnahmen Geltung zu verschaffen.

Bietet man Dienste wie Web- und Mailserver an, auf die aus dem Internet zugegriffen werden kann, so besteht darüber hinaus das Risiko, dass die Programme, die diese Dienste realisieren, fehlerhaft programmiert sind. Der Hauptteil der Schutzwirkung einer Firewall basiert ja darauf, den Zugriff auf Dienste zu verhindern. Hier aber ist genau dies erwünscht. Wenn also ein Zugriff auf einen We-

bserver dazu führt, dass der gesamte Rechner, auf dem der Dienst läuft, danach unter der Kontrolle eines Angreifers steht, so ist dieser Vorgang für eine Firewall in der Regel nicht von einem normalen lesen von Webseiten zu unterscheiden.

Die Firewall kann hier nur dagegen schützen, dass auf Dienste zugegriffen wird, die nicht öffentlich zugänglich sein sollen. Haben wir z.B. einen Webserver, der zusätzlich noch ein Fileserver ist, so kann eine Firewall verhindern, dass aus dem Internet auf die freigegebenen Dateien zugegriffen wird, sie kann aber nicht vor allen Angriffen auf den Webserver schützen. Dies ist der Grund, warum die öffentlichen Server oft in einem eigenen Netz (DMZ) zwar durch eine Firewall geschützt werden, dass lokale Netz aber noch einmal durch eine weitere Firewall abgetrennt ist. Man geht davon aus, dass eine Kompromittierung der Server grundsätzlich eine realistische Möglichkeit darstellt, weswegen man die Rechner im lokalen Netz nicht nur vor dem Internet, sondern auch von den eigenen öffentlichen Servern schützen muß.

2.5.3 Firewall Umgebungen

Eine Firewall, die nicht nur sich selbst schützt, sondern auch gleichzeitig als Router fungiert, wird als *dual homed host* bezeichnet, also ein Host, der in zwei Netzen zuhause ist. Mit dieser Technik ist es möglich, verschiedene Sicherheitsstufen zu realisieren, je nach Anforderung des zu schützenden Netzes. Im Folgenden haben wir die beiden wichtigsten Modelle dargestellt, die Bastion-Host-Firewall und die Demilitarisierte Zone (DMZ).

2.5.4 Die Bastion-Host-Firewall

Die einfachste - aber auch unsicherste - Form einer routenden Firewall ist die Bastion-Host-Firewall (Abbildung 2.5.4). Sie besteht grundsätzlich aus einem Rechner, der als *dual homed host* zwischen dem zu schützenden Netz auf der einen Seite und dem unsicheren Netz (Internet) auf der anderen Seite plaziert ist. Dieser Rechner routet Pakete von einem Netz ins andere. Jedes Paket, das durch diesen Rechner geroutet werden soll, muß die drei Regelketten der Firewall durchlaufen.

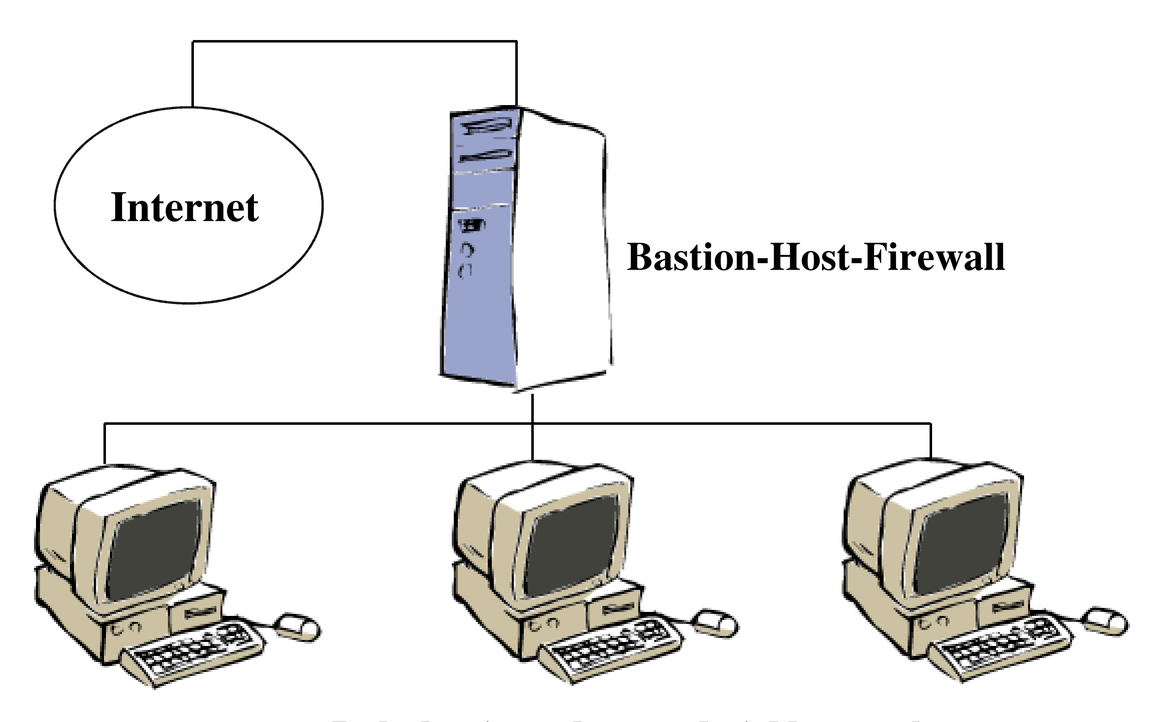


Abb. 2.5.4: Lokales (zu schützendes) Netzwerk

Der Begriff Bastion stammt aus dem Militärischen, er kann aber auch wirklich so verstanden werden. Eine Bastion ist eine Verteidigungseinrichtung, wenn sie fällt, dann gibt es keinen Schutz mehr. Diese Analogie können wir so ins Computernetz übernehmen. Wenn die Bastion-Host-Firewall geknackt werden würde, dann läge das gesamte Netz schutzlos da, der Angreifer, der die Firewall überwunden hätte, kann im LAN tun und lassen, was er will.

Trotzdem ist dieses Modell für die meisten kleinen Netze völlig ausreichend, wenn das zu schützende Netz nicht in großem Maß selbst Dienste im Internet anbieten soll.

2.5.5 Die Demilitarisierte Zone (DMZ)

Erheblich sicherer für große Netze ist das Modell der demilitarisierten Zone. Dieses Modell trennt das zu schützende Netz und das öffentlich zugängige Netz nochmal auf und arbeitet folgerichtig mit zwei Firewall-Rechnern (Abbildung 2.5.5).

Das lokale Netz, das die eigentlichen Arbeitsplätze verbindet, ist weiterhin das zu schützende Netz. Zwischen diesem lokalen Netz und der bösen großen weiten Welt des Internet liegt jetzt aber noch ein Netz, die demilitarisierte Zone. Wir haben zwei Firewall-Rechner, einmal der Bastion-Host, der das Internet mit der DMZ verbindet, und zum zweiten noch eine Firewall. Sie verbindet das zu schützende Netz mit der DMZ. Innerhalb der DMZ stehen jetzt die Rechner, die von außerhalb zugänglich sein sollen. Also etwa ein Webserver wie in unserem Projekt (siehe Abbildung 2.1.4 vom gesamten Netzwerk).

Diese Form der Firewall ist natürlich deutlich schwieriger zu verwirklichen, dafür aber auch wesentlich sicherer. Es existiert kein einziger Knackpunkt mehr, an dem alleine alle Sicherheit hängt. Der Ausfall bzw. die Eroberung eines Elements alleine führt noch nicht zur Unsicherheit des zu schützenden Netzes.

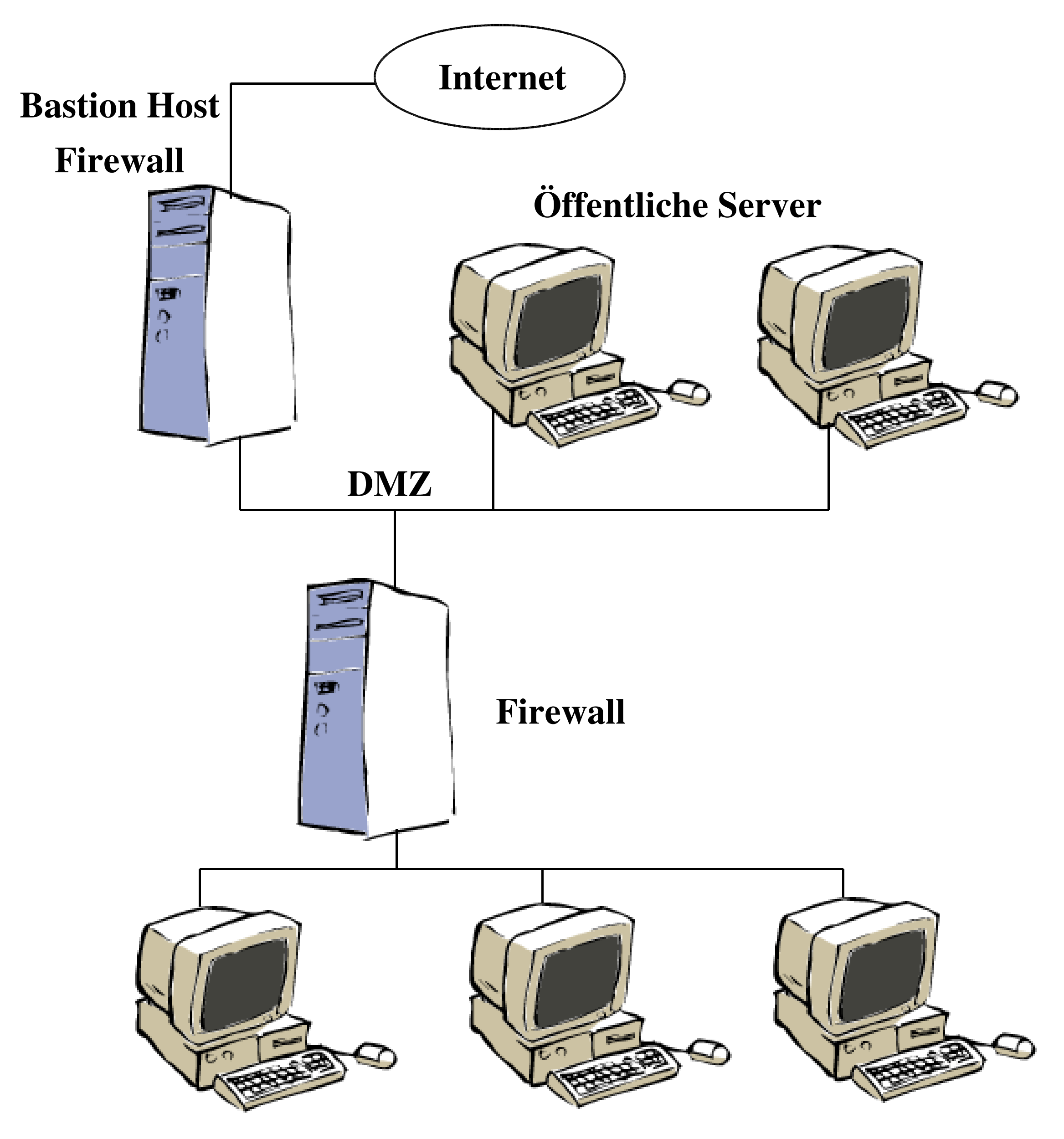


Abb. 2.5.5: Die DMZ

2.6 TCP/UDP

2.6.1 TCP

TCP, das Transmission Control Protocol, wird in RFC 793 beschrieben und benutzt IP, um Pakete zwischen Anwendungen zu transportieren und sicherzustellen, dass definierte Paketfolgen in der richtigen Reihenfolge beim Empfänger eintreffen, ohne dass Pakete doppelt gesendet oder ausgelassen werden.

Der Sinn einer Adressierung von Anwendungen wird klar,wenn man sich vergegenwärtigt, dass auf einem Rechner verschiedene Serverdienste gleichzeitig aktiv sein können. Ohne ein Konzept, wie man diese adressieren kann, müßte jeder Dienst jedes Paket entgegennehmen und dann entscheiden, ob es für ihn bestimmt ist.

TCP führt hierzu das Konzept der Portnummern ein. Sowohl Senderprozeß als auch Empfängerprozeß reservieren sich beim Betriebssystem eine Portnummer. Auf diese Weise können sowohl die Anfragen zugestellt als auch die Antworten an den Absender zurück befördert werden, da die Angabe von Serveradresse, Serverportnummer, Klientenadresse und Klientenportnummer eine Verbindung eindeutig beschreibt. Auch wenn ein Server auf demselben Port Anfragen von diversen Klientenprogrammen entgegennimmt, von denen einige vielleicht sogar denselben Rechner benutzen (wenn z.B. gleichzeitig Klienten verschiedener Hersteller eingesetzt werden), kann die Antwort immer eindeutig zugestellt werden (weil z.B. ein Klient Port 1024 zugewiesen bekam, während der andere Port 1025 benutzt).

Als Beispiel für das Portkonzept soll Abbildung 5.1 dienen. Hier haben wir einen HTTP- und FTP-Server, auf den mit drei Anwendungen vom selben Klienten aus zugegriffen wird.

Klient	***Server***	***Verbindung***
dummy:1024	wonderland:80	HTTP-Verbindung mit Netscape
dummy:1027	wonderland:21	FTP-Verbindung mit Netscape
dummy:1025	wonderland:80	HTTP-Verbindung mit kfm
dummy:1026	wonderland:21	FTP-Verbindung mit ncftp

Jedes Paket kann eindeutig zugestellt werden. Würde man eine der vier Angaben weglassen, so wäre dies nicht mehr möglich.

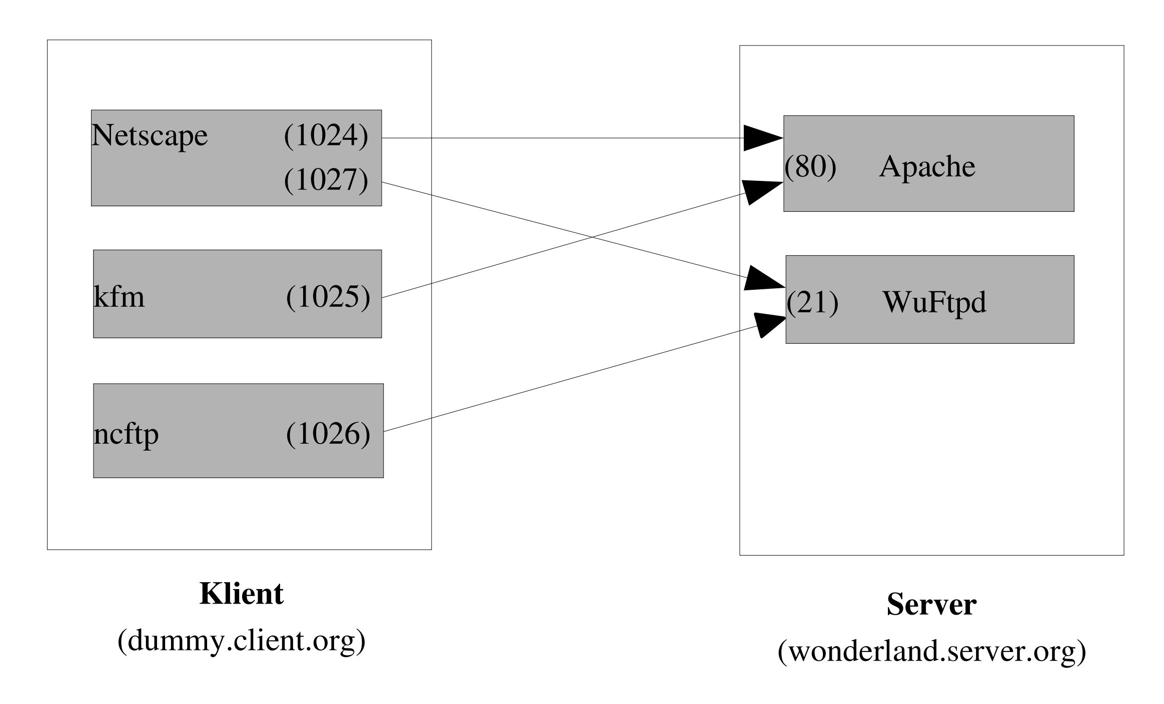


Abb. 2.6.1: Eindeutige Adressierung durch Ports

Im Internet sind bestimmte Portnummern fest vergeben. So benutzt DNS die Portnummern 53. Dabei gilt auf Unix-Systemen, dass Klienten in der Regel Portnummern größer 1023 benutzen, während die meisten Server auf Portnummern kleiner als 1024 auf Anfragen warten. Leider ist es so, dass sich nicht alle Server an diese Regel halten. So benutzt z.B. das X-Window-System standardmäßig die Portnummer 6000.

TCP soll darüber hinaus aber auch eine gewisse Verläßlichkeit der Übertragung sicherstellen. Dies ist nötig, da IP nur die Übertragung einzelner Pakete regelt, aber nicht deren Lieferung garantiert. So kann es passieren, dass Datenpakete verlorengehen, doppelt gesendet werden oder in der falschen Reihenfolge beim Empfänger ankommen.

Genau diese Probleme soll TCP beheben. Zu diesem Zweck werden alle gesendeten Bytes numeriert. Im Header jeden Paketes wird dabei die Nummer des ersten enthaltenen Datenbytes eingetragen *(Folgenummer).* Trifft nun ein Paket beim Empfänger ein, so überprüft dieser die Folgenummer des Paketes. Ist sie niedriger als der erwartete Wert, so wurde das Paket schon empfangen, und der Empfänger kann es entsorgen. Ist sie zu hoch, so muß ein Paket verlorengegangen sein. In diesem Fall wird der Empfänger den Absender darum bitten, noch einmal alle Pakete zu senden, deren Folgenummer größer ist als die des letzten Paketes mit einer erwarteten Folgenummer.Trifft

schließlich ein Paket mit einer korrekten Folgenummer ein, so sendet der Empfänger als Bestätigung die Nummer des nächsten erwarteten Paketes (Nummer des letzten empfangenen Datenbytes +1). Empfängt der Sender nach Ablauf einer gewissen Zeit keine Bestätigung, so wird er die unbestätigten Pakete noch einmal senden, da er annimmt, dass sie den Empfänger nicht erreicht haben. Dabei ist es durchaus möglich, dass mehrere Pakete gesendet werden, bevor eine Bestätigung erwartet wird.

Die Folgenummern werden nicht für jede Verbindung wieder auf null gesetzt. Statt dessen wird bei jedem Verbindungsaufbau auf beiden Seiten ein zufälliger Wert gewählt, der dann als Folgenummer für die eigenen Pakete dient. Dies hat den Vorteil, dass zufällig eintreffende Pakete aus früheren Verbindungen daran erkannt werden können, dass ihre Folgenummern außerhalb eines zulässigen Bereiches liegen.

Um der Gegenstelle mitzuteilen, welche Folgenummern man zu benutzen gedenkt, findet ein Verbindungsaufbau statt, der aus drei Paketen besteht. Im Header dieser Pakete sind dabei drei Felder für diese Darstellung interessant:

SEQ Folgenummer des Paketes (Sequence Number).

ACK Folgenummer, die von der Gegenstelle erwartet wird (Acknowledgement Number).

CTL Feld mit mehreren Bits, Flags genannt, die Kontrollinformationen signalisieren (Control Flags):

> **SYN** Dieses Paket gibt die neue Folgenummer des Senders bekannt. Wird nur beim Verbindungsaufbau benutzt (synchronize).
>
> **ACK** Dieses Paket bestätigt ein vorangegangenes Paket (acknowledge).
>
> **FIN** Der Sender wird keine weiteren Daten mehr schicken. Haben beide Seiten ein FIN gesendet, ist die Verbindung beendet (finish).
>
> **RST** Die Verbindung wird sofort beendet (reset).
>
> **PSH** Normalerweise sammelt der Kernel Daten erst einmal und gibt sie dann in größeren Blöcken an die Anwendung weiter. Dieses Flag soll das verhindern und bewirken, dass die Daten sofort weitergegeben werden (push). Außerdem garantiert das Setzen dieses Bits,dass das Paket bestätigt wird, unabhängig davon, ob dies normalerweise geschehen würde oder nicht. Das Flag wird unter Linux kaum verwendet.

URG Dient dazu, einen Teil der Daten im Paket als besonders eilig zu markieren (urgent). Ein zusätzliches Feld im TCP-Header namens >>Urgent Pointer<< gibt den Offset der betreffenden Daten innerhalb des Paketes an.

Der Verbindungsaufbau erfolgt nun in der Form:

	Rechner A				*Rechner B*
1.	⟶	SEQ=100		CTL=SYN	⟶
2.	⟵	SEQ=300	ACK=101	CTL=SYN,ACK	⟵
3.	⟶	SEQ=101	ACK=301	CTL=ACK	⟶

In diesem Beispiel beginnt Rechner A, indem er im ersten Schritt Rechner B mitteilt, dass er für seine Pakete mit der Folgenummer 100 beginnt. Rechner B bestätigt dies, indem er mitteilt, er erwarte nun ein Paket mit der Folgenummer 101. Zusätzlich gibt er an, er werde mit der Folgenummer 300 beginnen. Im dritten Schritt bestätigt Rechner A dies, indem er als Nummer des nächsten von ihm erwarteten Paketes die 301 angibt.

Merke: Ein ankommendes Paket, in dem das SYN-Bit gesetzt ist, das ACK-Bit aber nicht, weist darauf hin, dass eine Verbindung zu unserem Rechner aufgebaut werden soll.

2.6.2 TCP-Portbereiche

von	bis	Beschreibung
0	1023	Diese >>allgemein bekannten<< (>>well-known<<) Ports liegen in dem Bereich, den die meisten Systeme für privilegierte Prozesse vorgesehen haben. Die Internet Assigned Numbers Authority (IANA) kontrolliert die Abbildung von Servicenamen auf Portnummern in diesem Bereich.
1024	49151	Diese >>registrierten<< Portnummern liegen in einem Bereich, der bei den meisten Systemen von normalen Benutzerprozessen verwendet werden darf. Die IANA pflegt die Abbildung registrierter Servicenamen auf Portnummern in diesem Bereich, übt aber keine Kontrolle über die Zuweisung aus.
49152	65535	Die >>dynamischen<< oder >>privaten<< Portnummern, die nicht der Kontrolle oder Registrierung durch die IANA unterliegen.

2.6.3 UDP

UDP, das User Datagram Protocol, wird in RFC 768 beschrieben. Beim ihm handelt es sich um den >>kleinen Bruder<< von TCP. Auch hier werden Anwendungen über Ports adressiert. Die weitergehende Verbindungslogik von TCP mit seinen Folgenummern und Bestätigungen fehlt dagegen. Bei UDP werden einzelne Pakete, sogenannte *Datagramme,* abgeschickt in der Hoffnung, der Empfänger möge sie irgendwann irgendwie erhalten. Diese Art von Protokollen wird in der Fachwelt oft auch als >>Send and Pray<< bezeichnet.

Nun sollte man aber nicht den Fehler begehen, UDP für antiquiert und obsolet zu erachten. Tatsächlich gibt es eine Reihe von Anwendungen, bei denen UDP deutliche Vorteile gegenüber dem verbindungsorientierten TCP aufweist. Dies ist insbesondere im Multimedia-Bereich der Fall. Wenn Töne oder Filme in Echtzeit über das Netz übertragen werden sollen, stellt sich der Verwaltungsaufwand von TCP als spürbarer Nachteil heraus. Gehen z.B. bei einer Musikübertragung über das Internet einige wenige Pakete verloren, so können heutige Fehlerkorrekturverfahren die fehlenden Informationen genau genug aus den vorangegangenen Daten >>schätzen<<, dass es dem Hörer nicht auffällt.

Ein Warten auf das erneute Senden von Daten würde zu Pausen führen und den Hörgenuß empfindlich beeinträchtigen.

Andere Anwendungen von UDP betreffen Fälle, in denen eine Anfrage und die dazugehörige Antwort in ein Paket passen, womit die drei Pakete für den Verbindungsaufbau zu einem spürbaren Overhead werden. Dies war z.B. der Grund , warum DNS für kleinere Anfragen, die den Großteil der Fälle im normalen Betrieb ausmachen, UDP statt TCP benutzt. Schließlich existieren noch Fälle, in denen zwar Informationen verschickt werden, jedoch keine Antwort erwartet wird. Hier wird z.T. nicht gewünscht, dass zusätzliche Systemlast erzeugt wird, indem auf eine Empfangsbestätigung gewartet wird, die die sendende Anwendung nicht interessiert. Ein Beispiel hierfür ist das schreiben von Systemprotokollen über das Netzwerk. Hier werden Fehler- und Statusmeldungen statt sie in die lokale Protokolldatei zu schreiben, über das Netz an einen dedizierten Protokollierungsrechner geschickt. Das dabei verwendete Syslog-Protokoll sendet UDP-Pakete an Port 514 des Protokollierungsrechners, ohne dass jemals eine Antwort erfolgt.

2.6.4 Gängige TCP- (und UDP-) Portnummern

Port	Name	UDP	TCP	Beschreibung
7	echo	•	•	Echo Protocol (RFC 862)
9	discard	•	•	Discard Protocol (RFC 863)
13	daytime	•	•	Daytime Protocol (RFC 867)
19	chargen		•	Character Generator Protocol (RFC 864)
20	ftp-data		•	File Transfer Protocol (Data Stream)
21	ftp		•	File Transfer Protocol (Control Stream)
22	ssh		•	Secure Shell
23	telnet		•	Telnet Protocol (RFC854)
25	smtp		•	Simple Mail Transfer Protocol (SMTP)
37	time, timeserver	•	•	Time Protocol (RFC 868)
53	domain	•	•	Domain Name Service (DNS)
67	bootps	•	•	BOOTP-Server
68	bootpc	•	•	BOOTP-Client
69	tftp	•	•	Trivial File Transfer Protocol (TFTP)
80	http		•	Hypertext Transfer Protocol (HTTP)
109	pop2		•	Post Office Protocol (POP), Version 2
110	pop3		•	Post Office Protocol (POP), Version 3
111	sunrpc, portmapper	•	•	RPC Port Mapper (RFC 1050)
119	nntp		•	Network News Transfer Protocol (NNTP)
123	ntp	•		Network Time Protocol (NTP)
135		•	•	Microsoft: DHCP-Manager, WINS-Replikation, Exchange-Administrator, RPC für Exchange

Port	Name	UDP	TCP	Beschreibung
137		•	•	Microsoft: Browsing, WINS-Replikation
138		•		Microsoft: Browsing, Directory-Replikation
139			•	Microsoft: Filesharing (CIFS/SMB) und Print-Service, Directory-Replikation, Event Viewer, Logon Sequence, Performance Monitor
143	imap		•	Internet Mail Access Protocol (IMAP)
161	snmp	•		Simple Network Management Protocol (SNMP)
179	bgp	•	•	Border Gateway Protocol (BGP)
194	irc		•	Internet Relay Chat (IRC)
389	ldap		•	Lightweight Directory Access Protocol (LDAP)
443	https		•	HTTP über SSL
515	printer		•	Unix Print-Spooler
563	nntps		•	NNTP über SSL
631	ipp		•	Internet Printing Protocol (IPP)
636	ldaps		•	LDAP über SSL
873	rsync		•	Rsync (siehe *http://rsync.samba.org*)
993	imaps		•	IMAP über SSL
995	pop3s		•	POP Version 3 über SSL
1494			•	Microsoft: ICA (CITRIX)
2049	nfs, nfsd	•	•	Network File System (NFS)
3389			•	Microsoft: RDP (Remote Desktop Protocol)

2.7 NAT

2.7.1 Was ist Network Adress Translation (NAT)?

Gewöhnlich reisen Pakete in einem Netzwerk von ihrer Quelle (z.B. eigener Computer) zu ihrem Ziel (z.B. www.bbs.fh-wilhelmshaven.de) durch viele verschiedene Links. Nehmen wir an es wären 10 verschiedene Links bis zum Ziel. Keiner dieser Links verändert das Paket wirklich, sie schicken es einfach weiter.

Wenn einer dieser Links NAT machen würde, dann würde er die Quelle oder das Ziel des Pakets verändern, wenn es eintrifft. Wie man sich vorstellen kann, wurde das System entworfen, so zu arbeiten, also ist NAT immer etwas, was man mit Vorsicht behandeln sollte. Gewöhnlich wird sich der Link, der NAT macht, daran erinnern, wie er das Paket verändert hat, und wenn ein Antwortpaket aus der anderen Richtung kommt, wird er genau das Umgekehrte darauf anwenden, und so funktioniert es.

2.7.2 Warum sollte man NAT wollen?

Modemverbindung zum Internet

Die meisten Internetanbieter vergeben eine einzelne Adresse, wenn man sich bei ihnen einwählt. Man kann Pakete mit welcher Quell-Adresse auch immer verschicken, aber nur Pakete mit dieser Antwort-Adresse werden zu einem zurückkommen. Wenn man mehrere verschiedene Maschinen (so wie ein Heim-Netzwerk) benutzen will, um sich durch diesen Link mit dem Internet zu verbinden, wird man NAT brauchen.

Dies ist die heute am meisten verbreitete Art von NAT, gewöhnlich in der Linuxwelt als \`Masquerading\` bekannt. Man nennt dies auch SNAT, weil die Quell-Adresse des ersten Pakets geändert wird.

Mehrere Server

Manchmal möchte man ändern, wohin einkommende Pakete in einem Netzwerk gehen sollen. Oft ist das so, weil man(wie oben erwähnt) nur eine IP-Adresse hat, man möchte den Leuten aber die Möglichkeit geben, auch die Rechner hinter dem einen mit der \`echten\` IP-Adresse zu erreichen. Man kann das schaffen, wenn man das Ziel von einkommenden Paketen ändern kann.

Eine bekannte Variation dessen nennt sich \`load-sharing\`: Eine große Anzahl von Paketen wird über eine Reihe von Maschinen verändert, indem die Pakete \`aufgefächert\` werden. Diese Version von NAT wurde von früheren Linuxversionen Port-Forwarding genannt.

Transparente Proxies

Manchmal möchte man so tun, als ob jedes Paket, dass durch einen Linuxrechner geht, für ein Programm auf dem Linuxrechner selbst bestimmt ist. Dies wird für transparente Proxies verwendet. Ein Proxy ist ein Programm, dass zwischen einem Netzwerk und der Aussenwelt steht und die Kommunikation dazwischen regelt. Der transparente Teil kommt daher, weil das Netzwerk nicht einmal weiß, dass es mit einem Proxy redet, es sei denn natürlich, der Proxy funktioniert nicht.

Squid kann auf diese Art konfiguriert werden. Unter früheren Versionen von Linux hiess das Umleiten (redirection) oder auch transparentes Proxying.

2.8 Iptables

2.8.1 Die Idee

Mit dem Kernel 2.4.0 wurden Paketfilterung und Network Address Translation zu einem Konzept zusammengefaßt. Beide Mechanismen werden dabei mit dem Befehl iptables konfiguriert. Mit ihm werden Regeln formuliert, welche anhand der Header-Information eines Paketes entscheiden, was mit ihm geschehen soll.

Diese Regeln werden in Gruppen, sogenannten <<Chains>>, zusammengefaßt. Einige dieser Chains sind vordefiniert und werden nur für bestimmte Pakete zu Rate gezogen. So existiert z.B. eine Chain, die nur Filterregeln enthält, die auf Pakete angewendet werden, die von dem Rechner weitergeleitet werden und nicht für ihn selbst bestimmt sind. Andere Chains können vom Benutzer selbst definiert und aus Regeln in den Standard-Chains angesprungen werden.

Je nach Aufgaben werden die Chains in mehrere <<Tables>> zusammengefaßt. Der Table nat enthält Chains für die Network Address Translation, während der Table filter Chains zur Paketfilterung enthält. Die für uns interessanten Chains dieser beiden Tables sind in Abbildung 2.8.1 dargestellt. Die mit <<D-NAT>> und <<S-NAT>> markierten Chains stehen dabei im Table nat, die mit <<Filter>> bezeichneten im Table filter.

Trifft nun ein Paket über ein Netzwerk-Interface ein , so werden als erstes Regeln der Chain PREROUTING im Table nat ausgeführt. Hier können gegebenenfalls Zieladresse und -port geändert werden. Dies ist insbesondere praktisch, wenn man einen transparenten Proxy aufsetzen, d.h. alle Anfragen, die direkt an Server im Internet gestellt werden, auf einen Proxy auf der Firewall umleiten will.

Als nächstes wird eine Routing-Entscheidung getroffen. Falls das Paket für den Rechner selbst bestimmt ist, werden die Regeln in der Chain INPUT im Table filter angewendet. Hier wird entschieden, ob das Paket angenommen oder verworfen werden soll. Wird das Paket angenommen, so wird es nun an den Zielprozeß auf dem Rechner weitergeleitet.

Pakete, die nicht für den Rechner selbst bestimmt sind, sondern nur von ihm an ein anderes Netz weitergeleitet werden, werden statt dessen nach den Regeln in der Chain FORWARD im Table filter erlaubt oder verworfen.

Pakete, die von einem lokalen Prozeß ausgehen, werden einmal nach den Regeln der Chain OUTPUT im Table filter erlaubt oder verworfen. Hat ein Paket diese Prüfung bestanden, so besteht als nächstes die Möglichkeit, in der Chain OUTPUT im Table nat Zieladresse und -port zu ändern. Dies ist aber eine eher ungewöhnliche Maßnahme.

Für alle ausgehenden Paketen können schließlich mit den Regeln in der Chain POSTROUTING im Table nat Quelladresse und Port geändert werden. Damit kann z.B. Masquerading realisiert werden. Dabei werden alle Pakete so manipuliert, dass sie von der Firewall zu stammen scheinen. Dies ist insbesondere dann sinnvoll, wenn nur die Firewall selbst eine im Internet gültige Adresse besitzt. Darüber hinaus sorgt Masquerading dafür, daß die Struktur des internen Netzes aus dem Internet nicht sichtbar ist.

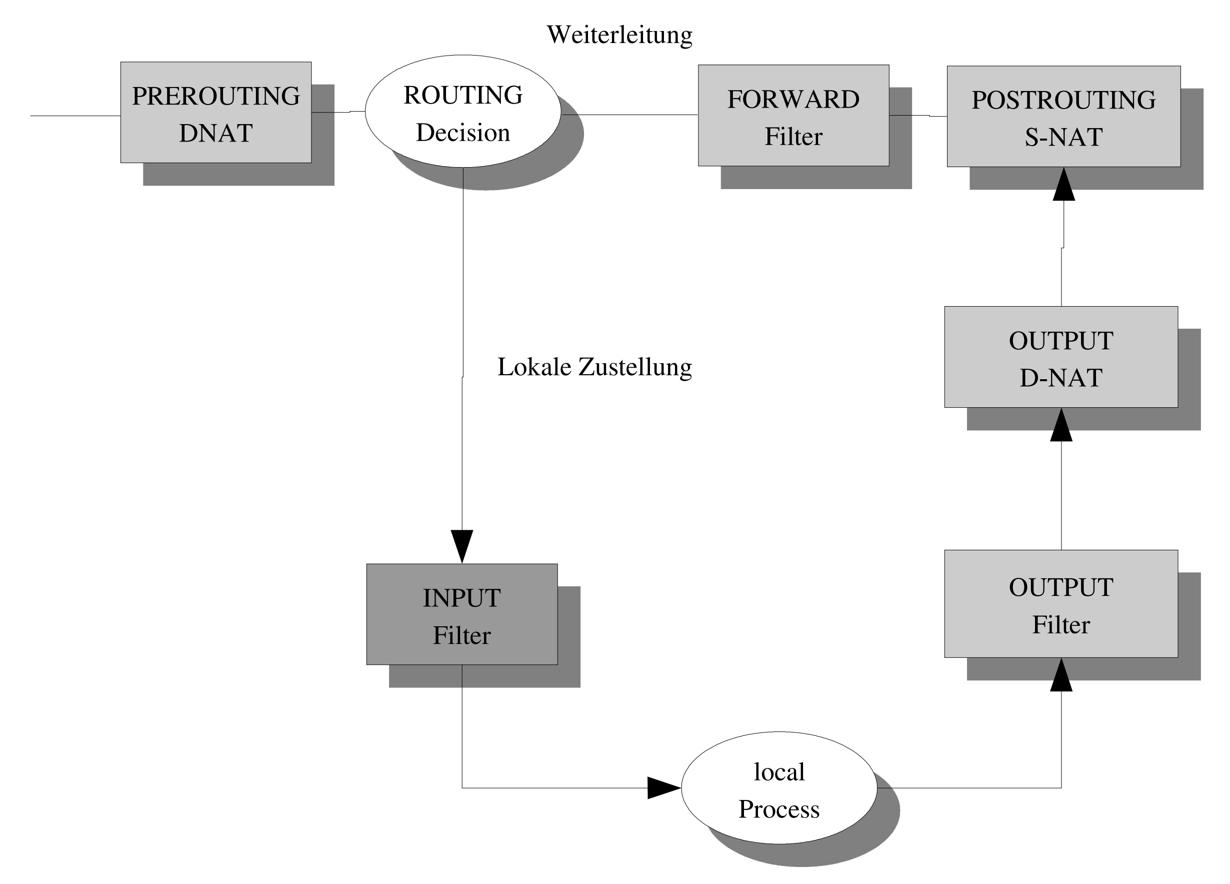


Abb. 2.8.1: Network Address Translation und Paketfilterung im Kernel 2.4

2.8.2 Policies

Jeder Standard-Chain im Table filter kann eine Grundregel mitgegeben werden, die Anwendung findet, wenn keine andere Regel greift. Dies nennt man eine Policy. Für iptables existieren die Policies:

ACCEPT Das Paket darf passieren.

DROP Das Paket wird verworfen.

Selbstdefinierte Chains besitzen keine Policies. Ist keine der Regeln der Chain anwendbar, so wird die nächste Regel derjenigen Chain untersucht, aus der in die selbstdefinierte Chain verzweigt wurde. Für das Einrichten einer Firewall erscheint die Policy ACCEPT nur in Sonderfällen sinnvoll. Üblicherweise wählt man beim Einrichten von Firewalls DROP nach dem Grundsatz:>>Was nicht erlaubt ist, ist verboten.<<

Die Syntax für das Erstellen einer Policy lautet:

iptables -P <Chain> <Policy>

2.8.3 Regeln

Eine Regel besteht aus Mustern und Aktionen. Die Muster legen fest, auf welche Pakete die Regel anzuwenden ist, und die Aktion definiert, was mit Paketen geschieht, wenn das Muster paßt.

Verwaltet werden die Regeln mit den folgenden Befehlen:

iptables [-t Table] -A chain Muster Aktion

iptables [-t Table] -I chain pos Muster Aktion

iptables [-t Table] -R chain pos Muster Aktion

iptables [-t Table] -D chain Muster Aktion

iptables [-t Table] -D chain pos

Dabei gilt:

-A Hängt eine Regel hinten an eine Chain an (append).

-I fügt eine Regel an einer Position *pos* ein (insert). Ist *pos* ausgelassen, so wird die Regel als erste eingefügt. Die Nummerierung der Regel beginnt mit 1.

-R ersetzt die Regel an Position *pos* (replace).

-D löscht eine Regel, die entweder über ihre Position *pos* oder ihre genaue Definition spezifiziert wurde (delete).

2.8.4 Tables

Bisher sind drei Tables definiert worden. Das Konzept ist aber auf Erweiterbarkeit ausgelegt, so daß weitere hinzukommen können. Auch hängt das Vorhandensein der Tables davon ab, wie der Kernel kompiliert wurde und welche Module geladen sind.

filter Dieser Table ist auch die Standardvorgabe, wenn kein Table angegeben wurde. Hier werden in den Chains INPUT, FORWARD und OUTPUT Regeln vorgegeben, die die Filterung von Paketen betreffen.

nat Dieser Table enthält in den Chains PREROUTING, OUTPUT und POSTROUTING die Regeln zur Manipulation von Quell- und Zieladressen sowie -ports.

mangle Spezielle Manipulationen an Paketen.

2.8.5 Muster

iptables ist auf Erweiterbarkeit ausgelegt. Nur wenige seiner Muster sind immer verfügbar. Weitere Muster können bei Bedarf hinzu geladen werden. Dies ist allerdings nur möglich, wenn Unterstützung für diese Erweiterungen als Modul oder Bestandteil des Kernels kompiliert wurde.

Wir wollen hier nur einen Blick auf die wichtigsten Muster werfen.

2.8.6 Standardmuster

Die folgenden Muster gehören zum Standardumfang. Jedes Muster kennt die Möglichkeit, mit >>!<< die Bedeutung des Musters in sein Gegenteil zu verkehren:

-p [!] **Protokoll** bezeichnet das verwendete Protokoll (tcp, udp, icmp oder eine nummerische Angabe).

-s [!] **Adresse[/Maske]** bezeichnet die Quelladresse (source). Als Maske kann entweder die Anzahl der zu betrachtenden Bits oder eine Maske angegeben werden. w.x.y.z/24 entspricht damit w.x.y.z/255.255.255.0

-d [!] **Adresse[/Maske]** bezeichnet die Zieladresse.

-i [!] **interface[+]** bezeichnet ein Interface,über das ein Paket empfangen wurde. Dabei ist es möglich, mit >>+<< alle Interfaces zu adressieren, die mit dem richtigen Namen anfangen. So würde -i eth+ sowohl auf eth0 als auch eth1 passen. Dieses Muster kann nur in INPUT-, FORWARD- und PREROUTING- Chains verwendet werden.

-o [!] **interface[+]** bezeichnet das Interface, über das das Paket gesendet werden wird. Dieses Muster kann nur in OUTPUT-, FORWARD- und POSTROUTING- Chains verwendet werden.

[!] -f spezifiziert Folgefragmente. Diese enthalten weder einen Quell- oder Zielport noch einen ICMP- Typ. Die normalen Regeln werden daher normalerweise nicht auf Folgefragmente passen. Wenn Sie allerdings Network Address Translation oder Connection Tracking benutzen, brauchen Sie sich keine Gedanken um Fragmente zu machen. In diesen Fällen werden Pakete sowieso zusammengefügt, bevor die Paketfilterregeln angewendet werden.

2.8.7 Aktionen

Ist ein Muster spezifiziert, gilt es, eine Aktion festzulegen, die von ihm ausgelöst werden soll. Dies geschieht mit

-j Target [Optionen]

Target kann hierbei u. a. eine der oben benannten Policies (ACCEPT, DROP) oder eine benutzerdefinierte Chain sein. Darüber hinaus existieren noch spezielle Targets, von denen hier nur die wichtigsten beschrieben werden sollen.

RETURN Das Paket >> fällt aus der Chain <<. D.h., es wird an die vorhergehende Chain zurückgereicht. Ist dies nicht möglich, weil es sich um die Regel einer Standard-Chain handelt, so tritt die Policy der Chain in Kraft.

LOG Das Paket wird im Systemprotokoll vermerkt. Dieses Target kennt mehrere Optionen, die die Protokollierung beeinflussen:

--log-level ***Level*** gibt die Priorität der Meldung an.

--log-prefix ***Prefix*** bezeichnet ein Präfix von bis zu 14 Zeichen, das der Protokollmeldung vorangestellt wird.

--log-tcp-sequence bewirkt die Protokollierung der TCP- Folgenummer. Dies kann ein Sicherheitsrisiko sein, wenn normale Benutzer das Systemprotokoll lesen können.

--log-tcp-options protokolliert die Optionen im TCP- Header.

--log-ip-options protokolliert die Optionen im TCP- Header.

2.8.8 REJECT

Das Paket wird verworfen, und es wird eine ICMP- Fehlermeldung an den Sender geschickt. Dieses Target kann nur im Table *filter* verwendet werden.

--reject-with Typ gibt die Fehlermeldung an, die gesendet wird. Möglich sind *icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreachable, icmp-net-prohibited* oder *icmp-host-prohibited.* Ohne diese Option wird *icmp-port-unreachable* verwendet. Als Reaktion auf ein Ping-Paket kann auch *echo-reply* verwendet werden. Für TCP-Pakete kann in der Chain INPUT oder einer eigenen Chain, in die aus INPUT verzweigt wurde, auch *tcp-reset* verwendet werden, wodurch ein TCP-Paket mit gesetztem RST-Flag gesendet wird.

2.8.9 SNAT

Dieses Target legt fest, dass das Paket weitergeleitet, die Quelladresse aber verändert wird. Dies geschieht nicht nur für das Paket selbst, sondern auch für alle nachfolgenden Pakete derselben Verbindung.

Das Target kann nur in der POSTROUTING- Chain des Tables *nat* und Chains, in die aus dieser verzweigt wurde, verwendet werden.

--to-source Addr[-Addr][:Port-Port] legt fest, welche Quelladresse im Paket eingetragen werden soll. Hier kann auch ein Adressbereich angegeben werden.

Unter Umständen ist es auch nötig, den Quellport des Pakets zu ändern. Wenn die Regel der Protokollangabe *-p tcp* oder *-p udp* enthält, kann hierzu ein Portbereich angegeben werden, andernfalls wird darauf geachtet, dass für Pakete, deren Quellport in einem der Bereiche 1-511, 512-1023, 1023-65535 liegt, dieser jeweils durch einen Quellport aus demselben Bereich ersetzt wird. Wo immer möglich, wird die Portangabe nicht geändert.

2.8.10 DNAT

ist das Gegenstück zu SNAT, hier wird die Zieladresse verändert. Gültig ist das Target nur in den PREROUTING- und OUTPUT- Chains des Tables *nat* sowie in daraus angesprungenen Chains.

--to-destination Addr[-Addr][:Port-Port] Hier kann sowohl ein Bereich von Adressen wie auch

Ports angegeben werden, aus denen eine neue Zielangabe generiert wird. Eine Portangabe ist dabei allerdings nur zulässig, wenn die Regel *-p tcp* oder *-p udp* enthält. Fehlt die Port angabe, so wird der Zielport nicht geändert.

2.8.11 MASQUERADE

Dieses Target legt fest, dass die Quelladresse des Pakets sowie nachfolgender Pakete derselben Verbindung in die Adresse des Interfaces geändert wird, über den es den Rechner verläßt. Wird der Zustand des Interfaces auf >>down<< geändert, so werden alle bestehenden Verbindungen >>vergessen<<. Dies ist insbesondere bei der Verwendung von dynamischen IP- Adressen sinnvoll. Beim nächsten Aktivieren des Interfaces wird ihm hier vielleicht schon eine ganz andere Adresse zugewiesen.

Das Target kann nur in Chains im Table *nat* verwendet werden. Hier auch nur in POSTROUTING sowie in Chains, in die aus POSTROUTING verzweigt wurde.

Wenn in der Regel *-p tcp* oder *-p udp* spezifiziert wurde, kann die folgende Option verwendet werden:

--to-ports Port[-Port] legt einen Portbereich fest, aus dem nötigenfalls ein Quellport gewählt wird. Es gelten dieselben Regeln wie bei SNAT.

2.8.12 REDIRECT

Dieses Target entspricht DNAT. Allerdings wird die Zieladresse auf den Rechner selbst gesetzt. Auf diese Weise können z.B. Anfragen an Server im Internet auf einen Proxy auf der Firewall umgeleitet werden.

Wenn in der Regel *-p tcp* oder *-p udp spezifiziert wurde, kann die folgende Option verwendet werden:*

--to-ports Port[-Port] legt einen Bereich von Ports fest, aus denen ein Zielport für das Paket gewählt wird.

2.9 Firewall Builder

!!! Info !!!

Dieses ist eine allgemeine Anleitung für den Fwbuilder, wonach jeder eine Firewall erstellen kann. Eine spezielle Anleitung im Bezug auf unsere beiden Projekt-Firewalls (mit allen verwendeten Regeln) finden Sie im Benutzer-Handbuch Kapitel 3.

2.9.1 Was ist der Fwbuilder ?

Firewall Builder (Fwbuilder) ist ein graphisches Konfigurationstool zum Erstellen und Verwalten von Firewall-Konfigurationen. Administratoren verwalten Netzwerk-Objekte wie beispielsweise Firewall-Rechner, Netzwerke, Hosts und Services. Per Drag & Drop können daraus Firewall-Policies erstellt und verändert werden. Die aktuellste Version für unser Projekt ist die Version 2.0.6. Damit werden über die verfügbaren so genannten Policy Compiler iptables, ipfilter, OpenBSD PF und Cisco PIX (kostenpflichtig) unterstützt. Die Policy Compiler übersetzen die Policies in ein für die Zielplattform verständliches Format. Mit einer zentralen Konfiguration können somit verschiedene Firewall-Typen verwaltet werden. Die Datenerhaltung erfolgt in einer XML-Datei.

2.9.2 Installation

Über die Homepage von Fwbuilder (http://www.fwbuilder.org) kann man sich den Quellcode laden und selber übersetzen. Andererseits stellt die Homepage eine Reihe vorkompilierter Binärpakete, so beispielsweise für Fedora Core, Mandrake, Red Hat und Suse zum Download bereit. Es werden die API-Bibliothek libfwbuilder, Fwbuilder selber sowie mindestens ein Modul , das die Erstellung der Regeln für die Zielplattform unterstützt, benötigt.

Für unser Projekt (Firewall mit Iptables) haben wir folgende RPM-Pakete installiert:

fwbuilder-2.0.6-1.fdr2.i386.rpm ---> Firewall Builder GUI

libfwbuilder-2.0.6-1.fdr2.i386.rpm ---> Firewall Builder API Libary

fwbuilder-ipt-2.0.6-1.fdr2.i386.rpm ---> Teile von der GUI und Policy Compiler für Iptables

Das Anwender-Handbuch, das ebenfalls über die Homepage zu erhalten ist, bezieht sich zwar auf eine ältere Version, beschreibt aber ausführlich die verschiedenen Einstellmöglichkeiten, wenngleich die neue Version doch bei der Oberfläche etwas anders aussieht. Dennoch sind die meisten beschriebenen Funktionen vorhanden. Es ist wie auch die Homepage nur in englischer Sprache erhältlich.

2.9.3 Allgemeines (Legende)

Bevor wir aber mit der Erstellung einer Firewall beginnen, eine kurze Legende für hier verwendete Begriffe. Siehe dazu Abbildung 2.9.1:

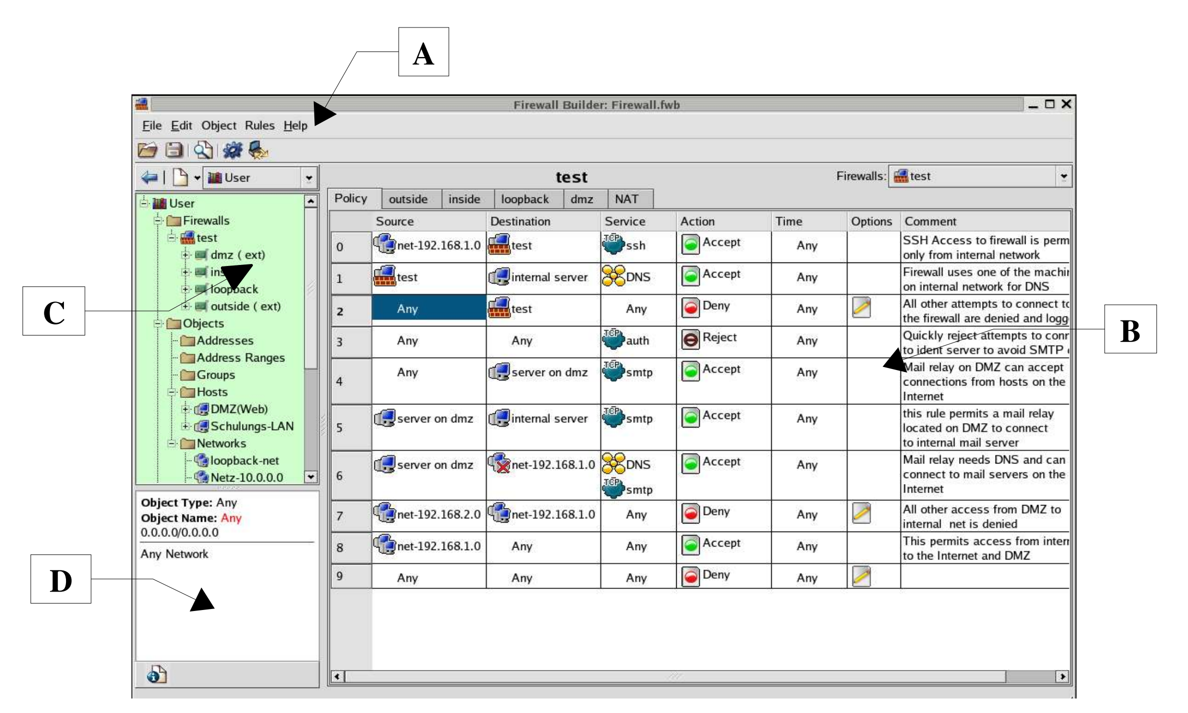


Abb. 2.9.1: Legende Fwbuilder

Die graphische Oberfläche von Fwbuilder teilt sich in zwei Bereiche (vergl. Abbildung 2.9.1). Auf der linken Seite (C) finden wir die Objekte, die in Bibliotheken organisiert sind. Unterhalb der Objekt-Struktur (D) finden wir abhängig von unseren Einstellungen Informationen zu dem ausgewählten Objekt. Auf der rechten Seite (B) stehen die Policies, die für die aktive Firewall definiert wurden. Über die Menüleiste (A) lassen sich z.B. Regeln einfügen, compilieren, Objekte erstellen usw.

Die Organisation von Netzwerk-Objekten erfolgt in Bibliotheken. Wir können auf bereits vorhandene Objekte der Standard-Bibliothek zugreifen. Eigene Objekte definieren wir in der Bibliothek USER. Dieser Name kann noch verändert werden und darüber hinaus können noch weitere angelegt werden.

Vor dem Erstellen von Regeln ist jedoch etwas Vorarbeit notwendig. Die mit gelieferten Bibliotheken definieren bereits eine Reihe von Objekten, die der Benutzer nutzen kann. Dabei sind vor allem die Objekte des Typs SERVICES sehr hilfreich, denn sie definieren bereits die gängigsten Dienste, die ansonsten manuell erstellt werden müssten. Alle fehlenden Objekte (z.B. Host-Objekte) müssen vor dem erstellen von Regeln definiert werden. Wie viel wir erfassen müssen hängt davon ab, wie fein die Regeln erstellt werden sollen und was wir aus den Standard-Bibliotheken bereits verwenden können.

2.9.4 Basiskonfiguration

Bevor wir den Firewall Builder das erste Mal starten, legen wir in */etc* als künftiges Arbeitsverzeichnis für die Applikation das Directory */etc/firewall* an. Dort sollen später die Konfigurationsdatei des Tools sowie die erstellten Firewall-Regeln lagern.

Jetzt starten wir den Firewall-Builder mit der Eingabe fwbuilder.



Abb. 2.9.2: Firewall Builder 2.0.6

Das Fenster (Abbildung 2.9.2) erscheint auf dem Desktop. Hier wählen wir Create new project file beim ersten Start aus. Ein weiteres Fenster öffnet sich (Abbildung 2.9.3)

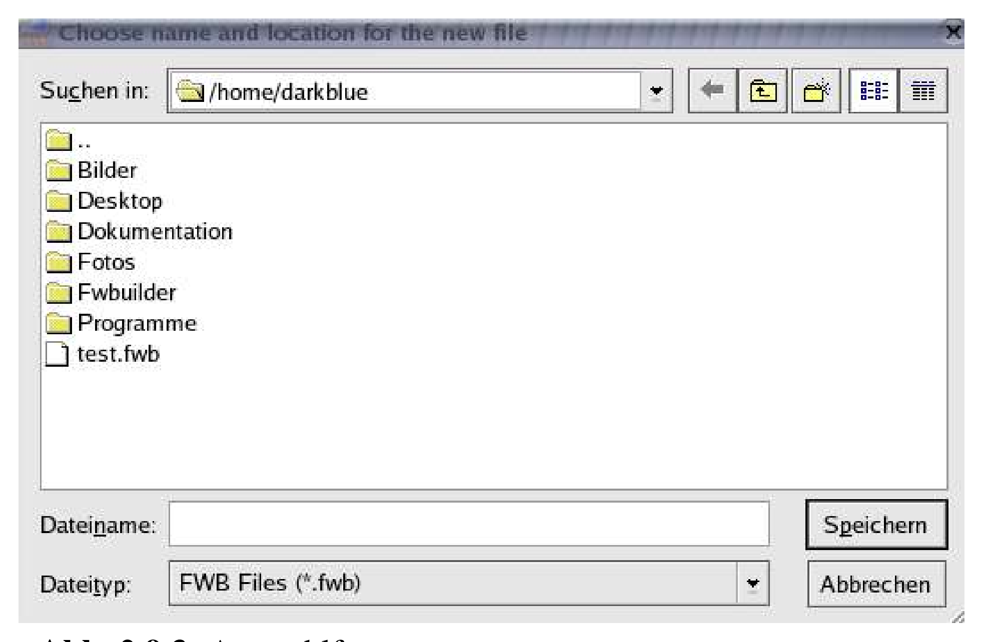


Abb. 2.9.3: Auswahlfenster

Dort wird der Name für die Konfiguration Datei Ihrer Firewall eingegeben mit der Endung .fwb und im XML Format dann gespeichert. Danach schließt sich dieses Fenster automatisch und Sie betätigen die Schaltfläche „weiter“ (Abbildung 2.9.2). Jetzt öffnet sich die Fwbuilder GUI (Abbildung 2.9.4) und Sie können mit der eigentlichen Arbeit beginnen.

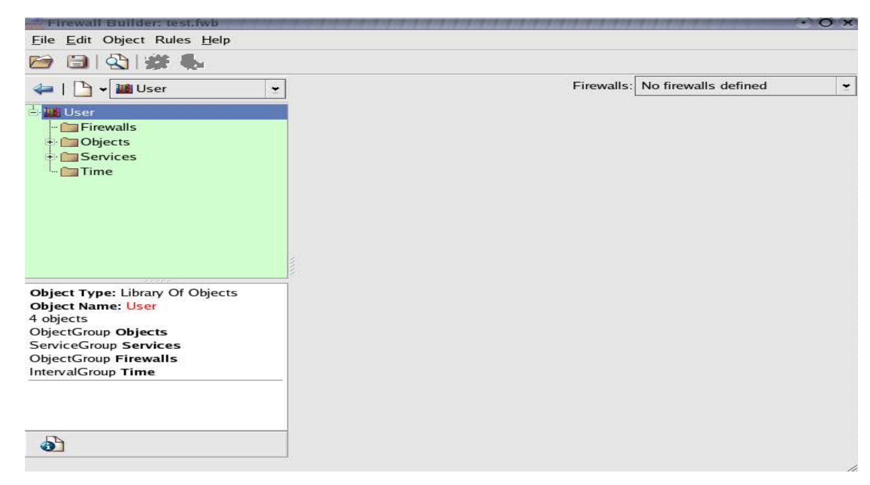


Abb. 2.9.4: Fwbuilder GUI

2.9.5 Firewall-Objekt erstellen

Nachdem wir ermittelt haben, welche Objekte fehlen, können wir uns daran machen, diese zu erfassen. Da ein Firewall-Rechner auf jeden Fall noch angelegt werden muss, machen wir dieses zuerst. Wir markieren im Objekt-Baum den Ordner FIREWALLS und wählen über das Kontextmenü der rechten Maustaste den Punkt NEW FIREWALL aus.

Beim Anlegen einer neuen Firewall geben wir dem Objekt zunächst einen Namen. Dabei sollte es sich um den Rechnernamen handeln. Dann legen wir fest, um welchen Firewall-Typen und welches Betriebssystem es sich handelt. Nun können wir noch auswählen, ob wir vorkonfigurierte Firewall-Objekte verwenden möchten. In diesem Fall bekommen wir eine Auswahl vordefinierter Firewall-Konfigurationen angezeigt (Abbildung 2.9.5).

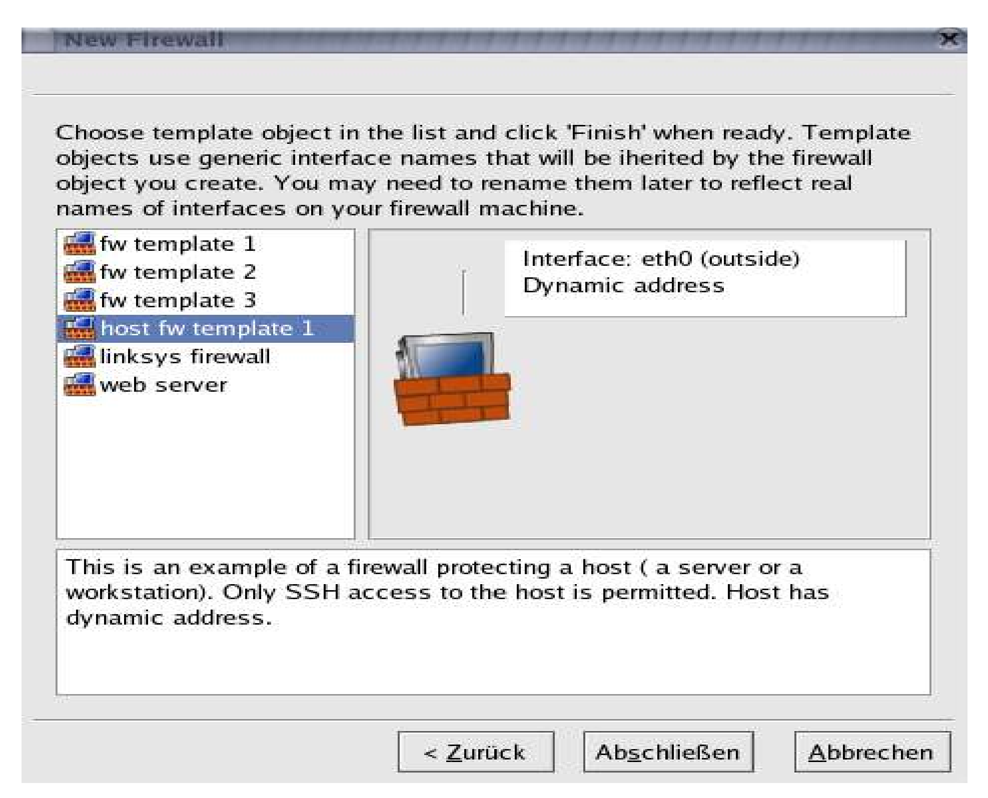


Abb. 2.9.5: Vordefinierte Firewall-Konfigurationen

Ein Beispiel einer solchen Konfiguration ist ein Rechner mit zwei Netzwerk-Karten, wobei das erste Interface mit dynamischer Adresse zum externen und die andere mit fester Adresse (192.168.1.1) zum internen Netzwerk gehört. Der ausgehende Netzwerkverkehr ist nicht beschränkt. Es werden aber nur Pakete rein gelassen, die zu einer existierenden Verbindung gehören. Nachdem wir ein Template ausgewählt haben, wird das neue Objekt in der Struktur angelegt und die für dieses Template hinterlegten Regeln in unsere Policies übernommen.

Verwenden wir kein Template, müssen wir die Netzwerk-Karten samt Adressen manuell anlegen. Anschließend haben wir die Möglichkeit, weitere Einstellungen zur Firewall und dem Betriebssystem vorzunehmen. Wenngleich die meisten Einstellung hier optional sind.

Wenn wir ein Template verwendet haben, sind nun schon ein paar Regeln für unser Firewall-Objekt zu sehen. Diese bilden eine gute Ausgangsbasis. Falls wir jedoch ein internes Netzwerk nicht im Adressraum 192.168.1.x betreiben, müssen einerseits einige Regeln und auch die Konfiguration der Netzwerk-Karte zum internen Netz verändert werden, denn die Templates basieren auf diesem Adress-Raum. Es ist leider beim Auswählen der Templates nicht möglich, deren Voreinstellungen zu verändern.

2.9.6 Andere Objekte

Neben den bereits beschriebenen Firewall-Objekten gibt es noch zahlreiche andere Objekt-Typen. So dienen Adressen-Objekte dazu, Adressen den Hostnamen zuzuordnen. Daneben dienen Adress-Bereiche dazu, das Netzwerk zu gruppieren, z.B. in ein Server- und ein Workstation-Segment.

GROUPS-Objekte dienen der logischen Zusammenfassung von Objekten. Falls wir verschiedene interne Netze betreiben, können wir die betreffenden Netzwerk-Objekte zu einer logischen Gruppe zusammenfassen, um damit allgemein gültige Regeln für sie zu definieren. Die Lesbarkeit der Policies verbessert sich dadurch erheblich.

Im Gegensatz zu Adressen-Objekten beschreiben Host-Objekte die betreffenden Computer etwas näher. Wie bei den Firewall-Objekten können wir aus Templates auswählen (Abbildung 2.9.6). Dabei handelt es sich um PCs mit einer oder zwei Netzwerk-Karten oder Router. Die Adressen liegen wie bei dem Firewall-Objekt im 192.168.1.0er- bzw. im 192.168.2.0er-Netz und müssen manuell angepasst werden. Neben den Templates ist auch hier eine manuelle Erfassung möglich. Der Vorteil des Host-Objektes liegt darin, dass man virtuelle Adressen Adressen an eine Netzwerk-Karte hängen kann.

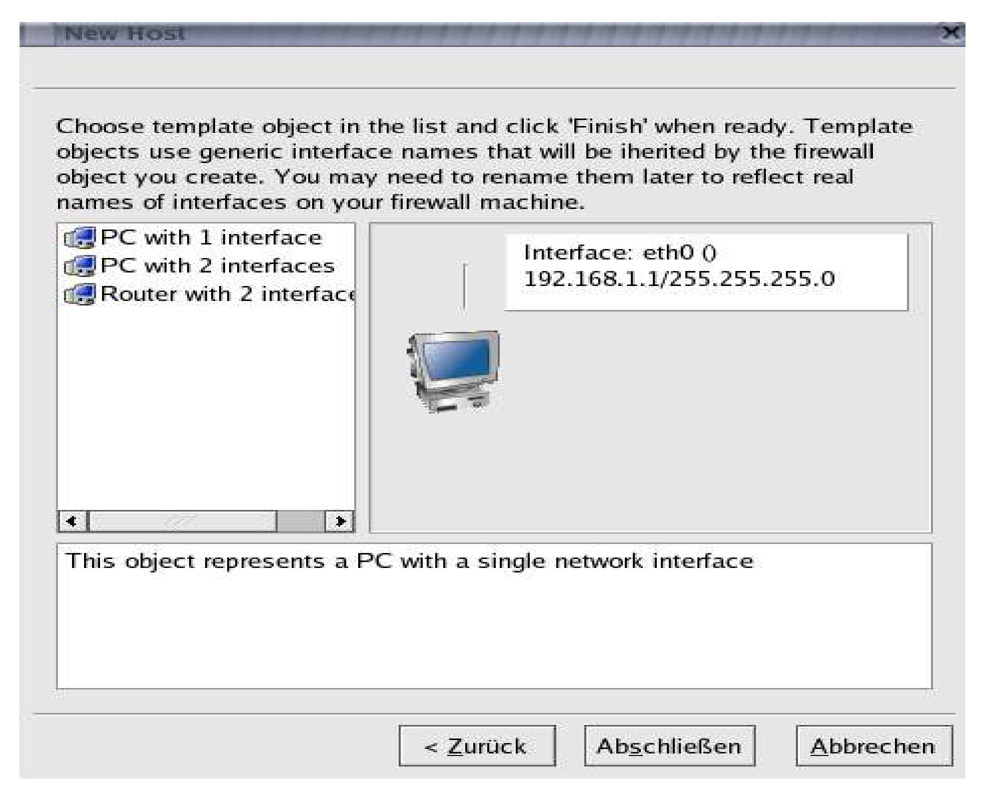


Abb. 2.9.6: Firewall Templates

Netzwerk-Objekte sind den Adressen-Objekten sehr ähnlich. Hierbei werden allerdings die Netzwerk-Adressen mit entsprechender Netzmaske definiert, beispielsweise 192.168.1.0/255.255.255.0.

Service-Objekte dienen zum Festlegen, wie Netzwerk-Pakete in den Regeln behandelt werden sollen. Hier können Objekte der Typen CUSTOM, GROUPS, ICMP, IP, TCP und UDP erfasst werden. Mit Hilfe der Standard-Objekte lässt sich recht einfach feststellen, wie diese einzusetzen sind.

Durch das Gruppieren lassen sich Service-Objekte ebenfalls logisch zusammenfassen. ein Beispiel aus der Standard-Bibliothek ist die Gruppe NETBIOS (Abbildung 2.9.7), die verschiedene UDP-Ports für Netbios zusammenfasst.

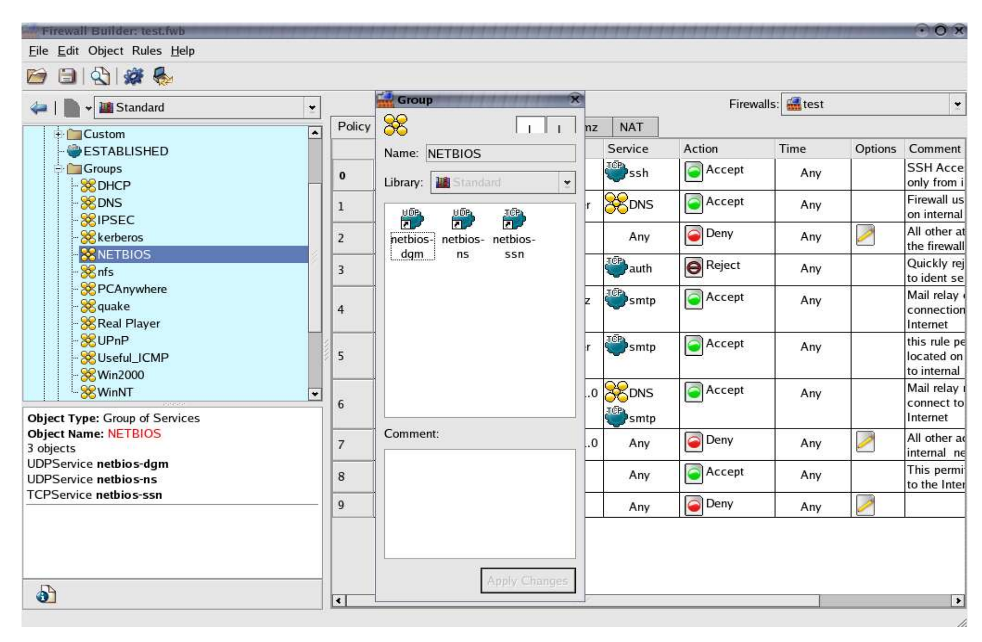


Abb. 2.9.7: Gruppe NETBIOS

Als Letztes gibt es noch den Objekt-Typ TIME (Abbildung 2.9.8). Mit diesen Objekten können wir Gültigkeitszeiträume in unseren Regeln bestimmen, beispielsweise, um bestimmte Dienste nur zu Bürozeiten zuzulassen.

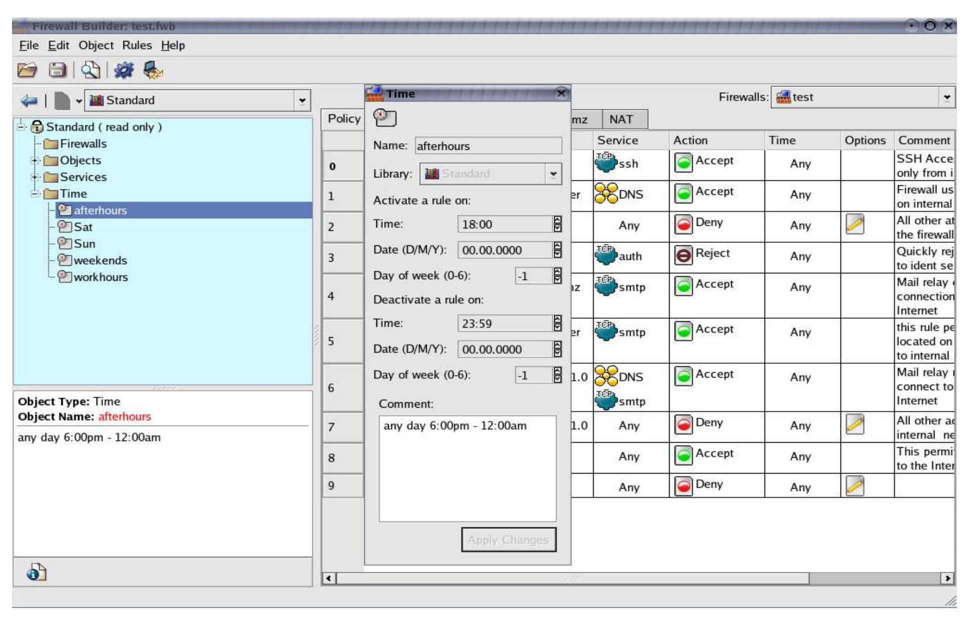


Abb. 2.9.8: Objekt TIME

2.9.7 Firewall-Policy erstellen

Nachdem wir uns in vorangegangenen Abschnitten mit dem Anlegen von Objekten beschäftigt haben, wollen wir uns jetzt mit dem Erstellen von Regeln befassen. Zunächst müssen wir die richtige Firewall aktivieren, falls mehrere vorhanden sein sollten. Rechts über den Regeln finden wir dazu eine Drop-Down-Box (Abbildung 2.9.9 (B)), die alle verfügbaren Firewall-Objekte enthält. Angezeigt wird zunächst das Formular namens POLICY (Abbildung 2.9.9 (A)), das die allgemeinen Regeln enthält.

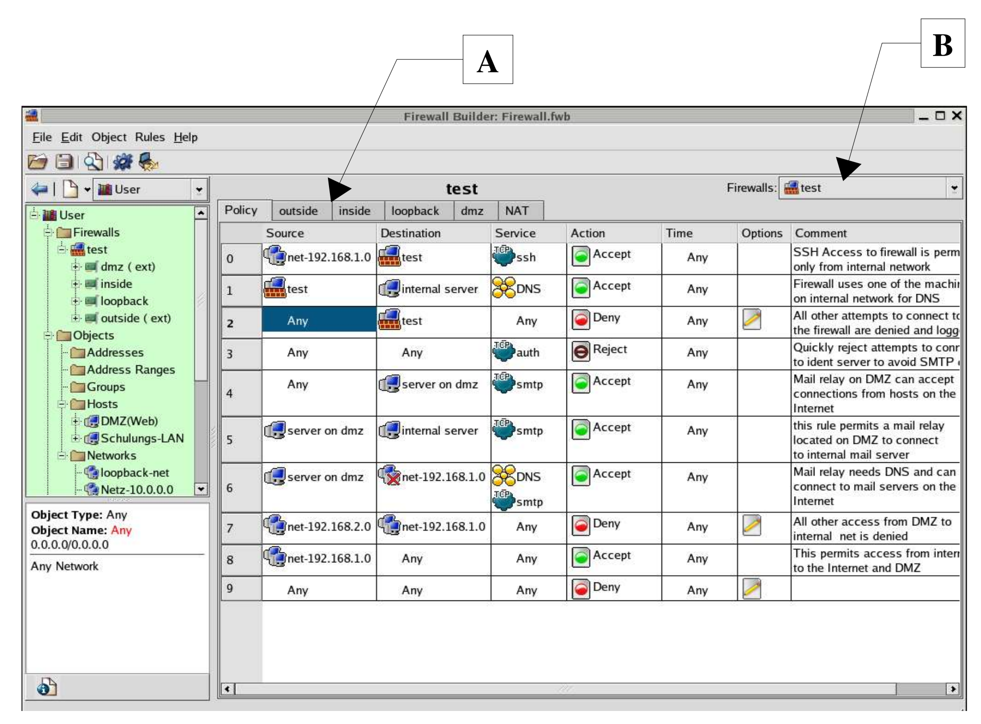


Abb. 2.9.9: Firewall-Policy erstellen

Daneben gibt es ein Formular für NAT-Regeln und eines für jedes Interface des Firewall Objektes. Im Normalfall werden die meisten Regeln im allgemeinen Teil erfasst, denn es sind meist nur wenige Interface-bezogene Regeln notwendig. Der Grund dafür ist, dass die globalen Regeln für jedes Paket geprüft werden.

Eine neue Regel erstellen wir, indem wir am linken Rand des Regel-Fensters bei der laufenden Nummer die rechte Maustaste drücken. Aus dem sich öffnenden Kontextmenü, wählen wir einfach INSERT RULE (Abbildung 2.9.10), um eine neue Regel vor der ausgewählten einzufügen.

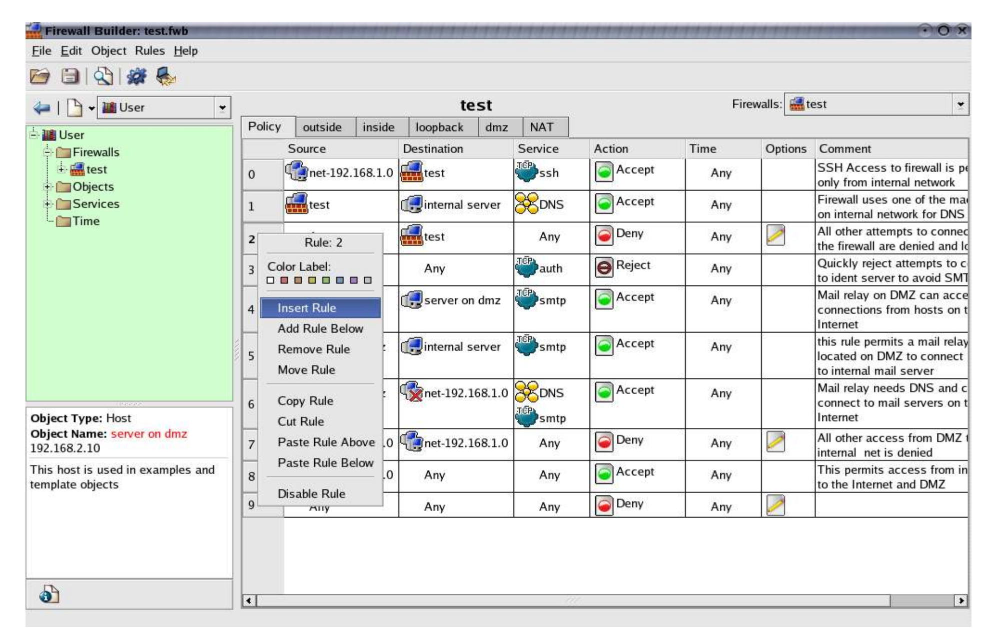


Abb. 2.9.10: Insert Rule

Mit ADD RULE BELOW erzeugen wir eine neue Zeile unterhalb der ausgewählten. Mit Hilfe von REMOVE RULE löschen wir Regeln, die wir nicht mehr benötigen. Falls wir die Reihenfolge von Regeln beeinflussen wollen, wählen wir MOVE RULE aus dem Kontextmenü. COPY RULE kopiert die ausgewählte Zeile und CUT RULE schneidet sie aus. Mit PASTE RULE ABOVE und PASTE RULE BELOW könne wir vorher ausgeschnittene oder kopierte Regeln vor bzw. nach der gewählten Regel einfügen. Wählen wir den Punkt DISABLE RULE, wird die betreffende Regel beim Kompilieren nicht berücksichtigt. Der Regel-Nummer wird ein rotes X vorangestellt. Rufen wir das Kontextmenü auf einer deaktivierten Regel auf, lautet dieser Menüpunkt ENABLE RULE, womit die Regel wieder für das Kompilieren aktiviert wird. Die gleichen Optionen sind auch über die Menüzeile über den Punkt RULES erreichbar.

Das Kontextmenü, das über die rechte Maustaste auf einem Objekt erscheint, sieht etwas anders aus. Über EDIT bearbeiten wir die Einstellung zu diesem Objekt. Mit COPY kopieren wir das Objekt und mit CUT schneiden wir es aus und legen es in die Zwischenablage. PASTE fügt das Objekt aus der Zwischenablage an die aktuelle Stelle ein. Das Auswählen von NEGATE negiert das aktuelle Objekt. Das Icon des Objektes erhält ein rotes Kreuz.

Mit INSERT RULE wurde nun eine leere Regel erzeugt, die alles verbietet. Als letzte Regel ist dies okay, aber wie bekommen wir nun vernünftige Regeln? Hier kommen nun die vorher angelegten und die Standard-Objekte zum Einsatz. Sie werden per Drag & Drop aus den Bibliotheken in die Regeln gezogen. Als SOURCE und DESTINATION können alle Adress-, Adressbereichs-, Gruppen-, Host- und Netzwerk- sowie Firewall-Objekte verwendet werden. Für SERVICE dementsprechend alle Objekte des Typs Service. Unter ACTION geben wir an, was geschehen soll, wenn die Regel zutrifft. Hier stehen Akzeptieren (ACCEPT), Verbieten (DENY) und Zurückweisen (REJECT) der getroffenen Pakete zur Auswahl. Verbotene Pakete werden stillschweigend verworfen, wohingegen für zurückgewiesene Pakete eine ICMP-Meldung zurückgegeben wird. Unter TIME können Objekte des gleichnamigen Typs verwendet werden. Über die OPTIONS-Spalte können wir beeinflussen, ob getroffene Pakete protokolliert werden sollen oder nicht. Die letzte Spalte dient der Erfassung eines Kommentares. In ihr sollte eine kurze Beschreibung der Regel erfasst werden, um später einen schnelleren Überblick zu bekommen. Der betreffende Inhalt wird auch in den generierten Code ausgegeben. Wollen wir ein Objekt von einer Regel entfernen, drücken wir einfach die ENTF-Taste oder wählen den entsprechenden Punkt aus dem Kontextmenü der rechten Maustaste.

2.9.8 Interface- und NAT-Regeln

Der Aufbau der Tabellen für die Interfaces ähnelt der des Policy-Formulars. Der einzige Unterschied liegt darin, dass es eine zusätzliche Spalte DIRECTION gibt (Abbildung 2.9.11). Hier wird festgelegt, auf welche Richtung, eingehend (inbound), ausgehend (outbound) oder in beide Richtungen, sich die Regel bezieht. Somit können beispielsweise Regeln erstellt werden, die alle Pakete beim externen Interface abweisen, die eine Ursprungsadresse aus dem internen Netzwerk haben (Anti Spoofing Regeln).

Über die Reiterkarte NAT (Abbildung 2.9.12)werden die Regeln für die Adress-Umsetzung getroffen, um beispielsweise eingehende Anfragen für Port 80 auf einen Webserver in der DMZ umzuleiten. Der Aufbau der Tabellen unterscheidet sich deutlich von denen der anderen. Hier wird festgelegt, wie die ursprünglichen Quell- und Ziel-Adressen sowie Services umgeschrieben werden sollen. Beispielsweise, um alle Adressen aus dem internen Netzwerk mit der Adresse des externen Interfaces zu versehen.

Bei allen Regeln können wir statt der bereits oben erwähnten Haupt-Objekte auch die Interface-Objekte von Host- und Firewall-Rechnern verwenden.

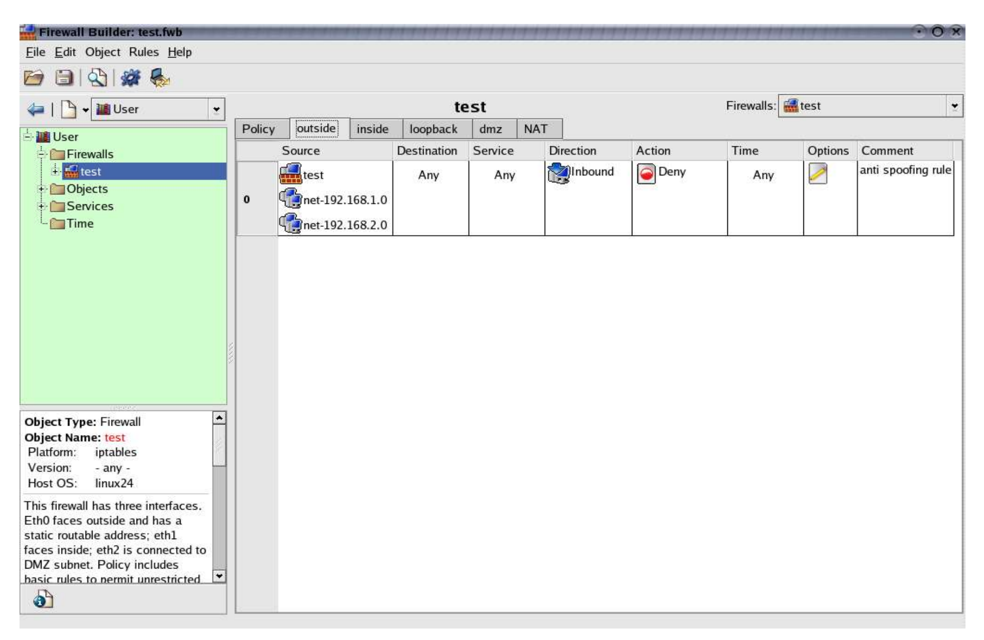


Abb. 2.9.11: zusätzliche Spalte DIRECTION

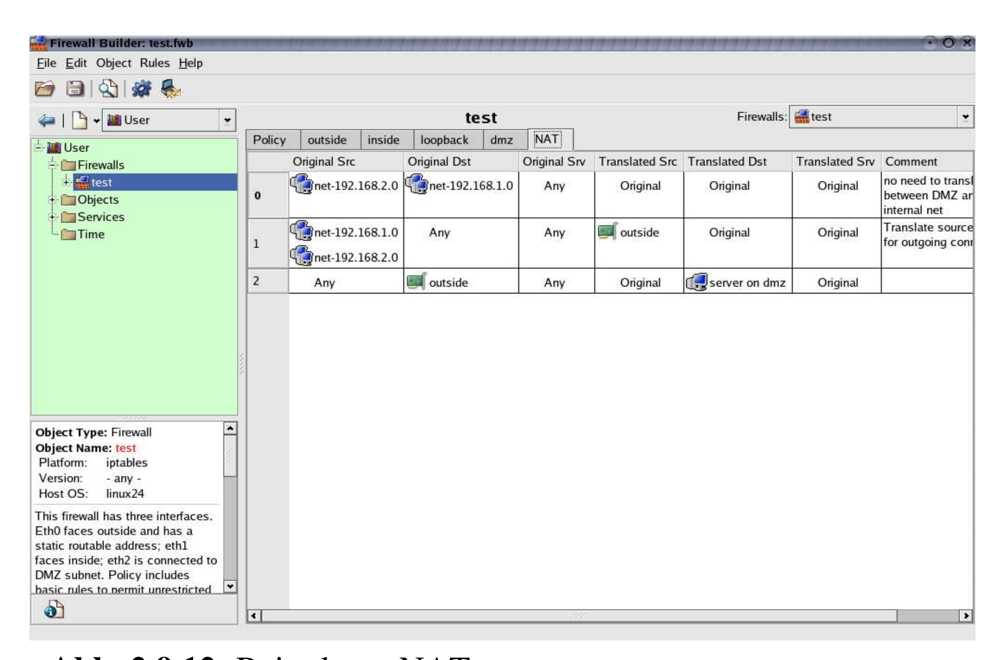


Abb. 2.9.12: Reiterkarte NAT

2.9.9 Übersetzung der Policies (Compilieren)

Jede Firewall verfügt, wie bereits oben erläutert, über eine globale Policy, eine für jedes Interface und eine NAT-Policy. Dabei gelten globale Regeln wie beschrieben für jedes Paket, das über die Firewall übertragen wird. Das ist unabhängig davon, über welches Interface sie ein- oder ausgehen. Beim Übersetzen der Regeln für die Zielplattform verarbeitet der Compiler zunächst die NAT-Regeln, dann die für die Interfaces und als Letztes die globalen Regeln. In dieser Reihenfolge werden die Zeilen des Ziel-Codes generiert. Den Code generieren wir über die Menüzeile unter RULES -> COMPILE oder über das Zahnrad in der Iconleiste darunter. Das generierte Skript heißt dabei immer wie der Name des Firewall-Objekts mit der Endung *.fw*.

Einen kurzen Ausschnitt für den generierten Code zeigt das Beispiel aus Abbildung 2.9.13:

```
#
# Rule 6 (global)
#
echo "Rule 6 (global)"
#
# Mail relay needs DNS and can
# connect to mail servers on the
# Internet
#
$IPTABLES -N Cid4268C6B9.0
$IPTABLES -A INPUT  -s 192.168.2.10  -d ! 192.168.1.0/24  -m state --state NEW  -j Cid4268C6B9.0
$IPTABLES -A Cid4268C6B9.0 -p tcp -m tcp  -m multiport  --dports 53,25  -j ACCEPT
$IPTABLES -A Cid4268C6B9.0 -p udp -m udp  --dport 53  -j ACCEPT
$IPTABLES -N Cid4268C6B9.1
$IPTABLES -A INPUT  -s 192.168.2.10  -d ! 192.168.1.0/24  -m state --state NEW  -j Cid4268C6B9.1
$IPTABLES -A Cid4268C6B9.1 -p tcp -m tcp  -m multiport  --dports 53,25  -j ACCEPT
$IPTABLES -A Cid4268C6B9.1 -p udp -m udp  --dport 53  -j ACCEPT
$IPTABLES -N Cid4268C6B9.2
$IPTABLES -A FORWARD  -s 192.168.2.10  -d ! 192.168.1.0/24  -m state --state NEW  -j Cid4268C6B9.2
$IPTABLES -A Cid4268C6B9.2 -p tcp -m tcp  -m multiport  --dports 53,25  -j ACCEPT
$IPTABLES -A Cid4268C6B9.2 -p udp -m udp  --dport 53  -j ACCEPT
#
# Rule 7 (global)
#
echo "Rule 7 (global)"
#
# All other access from DMZ to
# internal  net is denied
#
$IPTABLES -N RULE_7
$IPTABLES -A OUTPUT  -s 192.168.2.0/24  -d 192.168.1.0/24  -j RULE_7
$IPTABLES -A INPUT  -s 192.168.2.0/24  -d 192.168.1.0/24  -j RULE_7
$IPTABLES -A FORWARD  -s 192.168.2.0/24  -d 192.168.1.0/24  -j RULE_7
$IPTABLES -A RULE_7  -j LOG  --log-level info --log-prefix "RULE 7 -- DENY "
$IPTABLES -A RULE_7  -j DROP
#
# Rule 8 (global)
#
```

Abb. 2.9.13: Ausschnitt aus generiertem Code

2.9.10 Installation der Regeln

Die Installations-Routine starten wir über die Menüzeile unter RULES -> INSTALL bzw. über das Bild mit der Workstation und den beiden Pfeilen in der Iconleiste direkt neben dem Zahnrad. Falls es schon ein generiertes Firewall-Skript gibt, haben wir die Möglichkeit, dieses ohne Änderung zu verwenden oder auch neu zu kompilieren. Gibt es noch kein Skript, wird es vor der Installation erstellt. Während der Installation können wir den von uns voreingestellten Benutzer für den Transfer ändern. Das Kennwort für die Installation lässt sich aus Sicherheitsgründen nicht abspeichern und muss bei jeder Installation von neuem eingegeben werden. Handelt es sich nicht um root, sondern um einen User muß dieser über die entsprechenden Rechte verfügen, um die Regeln zu aktivieren.

2.9.11 Fazit

Wenngleich wir mit diesem Programm recht einfach unsere Regelsätze zusammenstellen und installieren konnten, sind dennoch Firewall-Kenntnisse für die Zielplattform (in unserem Fall iptables) notwendig. Die vordefinierten Objekte der Standard-Bibliothek bildeten eine gute Ausgangsbasis für unsere Regeln. Dennoch mussten einige Objekte manuell erfasst werden. Wenn die Objekt-Basis erst einmal geschaffen wurde, wird jeder, der schon manuell iptables-Regeln erstellt hat, die Einfachheit der Regel-Erstellung mit diesem Programm zu schätzen wissen. Dass beim Übersetzen der Regeln überlagernde Regeln erkannt werden, ist eine hilfreiche Funktion.

Insgesamt würden wir sagen: Wer die ersten Hürden mit diesem Programm genommen hat, der wird es nicht mehr missen wollen.

3 Fazit

Im Vorfeld konnte keiner von uns Erfahrungen im Bereich Firewall nachweisen. Die Projektarbeit erlaubte es uns nun in ein neues und sehr interessantes Gebiet vorzustossen. Wir hatten große Freude an der Konzeption und Entwicklung des Systems und haben dementsprechend auch viel Zeit und Herzblut investiert. Die Arbeit mit Tools wie dem Firewall Builder hat uns mit der Zeit dazu ermutigt weitere Open Source Lösungen in unsere Projektarbeit einfliessen zu lassen oder uns von entsprechenden Lösungsansätzen inspirieren zu lassen. Wir konnten feststellen, dass für (fast) jede kommerzielle Software ein ebenbürtiges Open Source Derivat existiert. Zwar sind diese nicht immer ausreichend dokumentiert, doch wenn man die Zeit hat und in den richtigen Foren „stöbert“ kommt man schon auf einen „grünen Zweig“.

Wir sind der Meinung, dass dieses Projekt nicht nur eine außergewöhnliche Herausforderung für uns darstellte, sondern, dass wir auch jede Menge gelernt haben (z.B. Teamfähigkeit), was uns auf den zukünftigen Berufsalltag als Techniker authentisch vorbereitet hat.

Außerdem besitzen wir nun endlich auch eine eigene Firewall, also ganz uneigennützig war diese Projektarbeit nicht.

Nun ja, dieses Loblied könnte hier unendlich fortgesetzt werden, aber wir denken hier ist der Ort und Zeitpunkt um allen Beteiligten ein großes Dankeschön auszusprechen.

Danke schön an alle Beteiligten !!!

4 Abbildungsverzeichnis

5 Wochenprotokolle

Wochenprotokolle vom 07. Februar 2005 bis 09. Mai 2005



5.1 Projektwoche 1

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 07. Februar 2005 bis 11. Februar 2005

Protokollant: Michael Bruckmann

<u>Teambesprechung:</u>

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Festlegung eines Projektleiters bzw. Ansprechpartner und Besprechung der allgemeinen Vorgehensweise während der Projektarbeit

Festlegung eines Ansprechpartners im Projektteam. Als Projektleiter wurde Michael Bruckmann gewählt. Informationen über unsere Projektarbeit und ihre Arbeitsbereiche wurden uns von unserem Projektleiter Herrn Appenzeller vermittelt

Projektbesprechung: Installation, Netzwerk, Tools, Anbindungen, Server usw.

Thema: Allgemeiner Informationsaustausch

- ☑ Festlegung der Informationsbeschaffung (Bücher Beschaffung, Internetrecherche)
- ☑ Verwendetes Betriebssystem auf dem Server wurde festgelegt
- ☑ Festlegung der Software (Zusatz)
- ☑ Allgemeine Aufgabenverteilung und Ablauf der Projektarbeit wurden besprochen

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.2 Projektwoche 2

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 14. Februar 2005 bis 18. Februar 2005

Protokollant: Olaf Plaggenborg

<u>**Teamarbeit:**</u>

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Inbetriebnahme des Servers und Installation

- ☑ Installation von Linux Fedora Core 2 im Raid Level 1 (Software Raid)
- ☑ Installation erfolgreich durchgeführt, aber nach dem Neustart bootet das System nicht mehr
- ☑ Bootloader Grub startet nicht
- ☑ Fehlersuche durchgeführt
- ☑ Bios Update auf Version 1.03 durchgeführt und Betriebssystem neuinstalliert
- ☑ System bootet, Bootloader Grub startet, System funktioniert
- ☑ Einlesen in das Thema „Iptables“
- ☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.3 Projektwoche 3

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 21. Februar 2005 bis 25. Februar 2005

Protokollant: Michael Bruckmann

<u>**Teamarbeit:**</u>

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Inbetriebnahme des Servers und Installation

- ☑ Konfiguration des Betriebssystems (Dienste, Software z.B. Webmin)
- ☑ Netzwerk einrichten
- ☑ Fehler! nur eine Netzwerkkarte wird erkannt, nicht alle drei Netzwerkkarten
- ☑ Fehlersuche durchgeführt
- ☑ Durch Einbau von drei verschiedenen Netzwerkkarten nacheinander Fehler behoben
- ☑ Einrichten des Netzwerkes mit Webmin volle Funktion des Netzwerkes
- ☑ Netzwerk-Routen eingerichtet
- ☑ Einlesen in das Thema „Iptables“
- ☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.4 Projektwoche 4

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 28. Februar 2005 bis 04. März 2005

Protokollant: Olaf Plaggenborg

<u>Teamarbeit:</u>

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Installation Firewall Builder

- ☑ Firewall Builder von der Internetseite runter geladen (GUI, Libary und Policy Compiler)
- ☑ Firewall Builder installiert und eingerichtet
- ☑ Englisches Handbuch heruntergeladen und ausgedruckt (Enthält leider nicht die neuste Version)
- ☑ Erste Schritte mit dem Firewall Builder anhand von vorhandenen Templates
- ☑ Einlesen in das Thema „Iptables“
- ☑ Einlesen in das Thema „Firewall-Architekturen“
- ☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.5 Projektwoche 5

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 07. März 2005 bis 11. März 2005

Protokollant: Michael Bruckmann

Teamarbeit:

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Arbeiten mit dem Firewall Builder

☑ Firewall Builder User Guide (Englisch) wichtige Texte übersetzt

☑ Internetrecherche zum Thema „Firewall", „Iptables" und „Firewall Builder"

☑ Erste Versuche einer Test Firewall auf dem Notebook

☑ Einlesen in das Thema „Iptables"

☑ Einlesen in das Thema „Firewall-Architekturen"

☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.6 Projektwoche 6

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 14. März 2005 bis 18. März 2005

Protokollant: Olaf Plaggenborg

Teamarbeit:

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Arbeiten mit dem Firewall Builder

- ☑ Angefangen Firewall auf dem Server einzurichten
- ☑ Notebooks für Testversuche eingerichtet
- ☑ Notebook 1 zum Webserver hergerichtet
- ☑ Notebook 2 zum simulieren der internen Netze vorbereitet
- ☑ Kommentare für den Firewall Builder eingetragen
- ☑ Versucht die Firewall zu compilieren, Fehler bei den NAT-Regeln
- ☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.7 Projektwoche 7

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 04. April 2005 bis 08. April 2005

Protokollant: Michael Bruckmann

<u>Teamarbeit:</u>

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Erstellen der Firewall (BBS-FIRE)

- ☑ Fehler in den NAT-Regeln gefunden und beseitigt
- ☑ Compilieren der gesamten Firewall (BBS-FIRE) funktioniert
- ☑ Installation der Firewall mit Hilfe vom Firewall Builder
- ☑ Besprechung wie man am besten einen Testaufbau durchführen könnte, um alle Funktionen der Firewall zu überprüfen
- ☑ Einarbeiten in diverse Tools zum testen der Firewall z.B. Nmap (manpage lesen usw.)
- ☑ Testaufbau mit Hilfe der vorbereiteten Notebooks durchgeführt
- ☑ Einlesen in das Thema „Iptables", „Firewall-Architekturen"
- ☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.8 Projektwoche 8

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 11. April 2005 bis 15. April 2005

Protokollant: Olaf Plaggenborg

<u>Teamarbeit:</u>

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Erstellen der Firewall (BBS-FIRE)

- ☑ Test der Firewall durchgeführt alles in Ordnung
- ☑ Webmin installiert und konfiguriert
- ☑ Zweite Betriebssystem-Installation auf dem Server durchgeführt, diesmal eine minimal Installation ohne Grafische Oberfläche
- ☑ Netzwerk bei dieser Installation eingerichtet
- ☑ NIC´s angepasst wie bei der grafischen Installation
- ☑ Webmin auch hier installiert
- ☑ Firewall versucht zu installieren, Problem keine grafische Oberfläche vorhanden
- ☑ Firewall von anderen Server mit dem Firewall Builder installiert funktioniert
- ☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.9 Projektwoche 9

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 18. April 2005 bis 22. April 2005

Protokollant: Michael Bruckmann

Teamarbeit:

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Erstellen der Firewall (SUPER-SERVER)

- ☑ Änderungen an der Firewall durchgeführt, Internes Netz mit neuen Anforderungen
- ☑ Firewall erfolgreich umgesetzt
- ☑ Erstellung der zweiten Firewall für den Super-Server (Projekt Terminal Server)
- ☑ Installation des Firewall Builder auf dem Super-Server
- ☑ Installation der angefertigten Firewall auf dem Super-Server
- ☑ Vernetzung aller drei Projekt-Server (BBS-FIRE, SUPER-SERVER und WEB-SERVER)
- ☑ Test der beiden Firewalls durchgeführt, kein gewünschter Erfolg im Zusammenspiel
- ☑ Fehlersuche
- ☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

5.10 Projektwoche 10

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 25. April 2005 bis 29. April 2005

Protokollant: Olaf Plaggenborg

<u>Teamarbeit:</u>

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Gesamt-TEST bzw. Fehlersuche

- ☑ Fehlersuche weiter durchgeführt
- ☑ Fehler behoben, Test in Ordnung
- ☑ Weitere Änderung, der Proxy-Server muss aus Sicherheitsgründen vom Terminal-Server umziehen auf den Firewall-Rechner, um die Rechte der Terminal-Clients einschränken zu können
- ☑ Einrichten des Proxy-Servers auf BBS-FIRE
- ☑ Umstellung der Firewall-Regeln auf beiden Servern
- ☑ Neuer Testaufbau, alle Funktionen testen
- ☑ Internetrecherche
- ☑ Dokumentation

Praktische Durchführung: Berufsbildenden Schulen Friedenstraße Wilhelmshaven

5.11 Projektwoche 11

Projekt: Aufbau einer collapsed Firewall mit Grenznetz und Bastion Host mit der Möglichkeit zur Remoteadministration

Protokoll vom: 02. Mai 2005 bis 06. Mai 2005

Protokollant: Michael Bruckmann

Teamarbeit:

Ort: Berufsbildende Schulen Friedenstraße Wilhelmshaven

Thema: Abschluß

☑ Gesamt Test durchgeführt alle Funktion sind gegeben

☑ Abschlußarbeiten durchgeführt

☑ Kommentare eingetragen

☑ Entgültige Firewall auf minimal Betriebssystem installiert

☑ Abschlußbesprechung

☑ Dokumentation

Praktische Durchführung: Berufsbildende Schulen Friedenstraße Wilhelmshaven

6 Anhang

6.1 Quellenverzeichnis

Literatur

Autor	Titel	Verlag
Wolfgang Barth	Das Firewall Buch	SuSE PRESS
Andreas G. Lessig	Linux Firewalls	O´REILLY
Gregor N. Purdy	Linux iptables	O´REILLY
Brian Ward	Der LINUX-Problemlöser	dpunkt.verlag
Dr. Rainer Hattenhauer	Red Hat & Fedora Linux	DATA BECKER

Internet

Titel	Internet-Adresse
Webmin	http://www.webmin.de
Firewall Builder	http://www.fwbuilder.org
Wikipedia Freie Enzyklopädie	http://de.wikipedia.org/wiki/Hauptseite